

# Edition Notices

Take Note!  
Before using this information and the product it supports, be sure to read the general information under [Special Notices](#).

## First Edition (January 1996)

This edition applies to IBM OS/2 Warp Version 3.0.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback has been included in this online book. Comments may also be addressed to:

IBM Corporation, International Technical Support Organization  
Dept. JLPC Building 014-1 Internal Zip 5220  
1000 NW 51st Street  
Boca Raton, Florida 33431-1328.

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

**Copyright International Business Machines Corporation 1996. All rights reserved.**

Note to U.S. Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

-----

# Special Notices

This publication is intended to help service personnel, system programmers and software developers to understand the concepts and application of debugging techniques. The information in this publication is intended as a supplement to already published specifications of any programming interfaces that are provided by IBM Warp OS/2 Version 3. See the PUBLICATIONS section of the IBM Programming Announcement for IBM Warp OS/2 Version 3 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

-----

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM  
Micro Channel  
OS/2  
Operating System/2  
Presentation Manager  
WIN-OS/2  
Workplace Shell  
XGA

The following terms are trademarks of other companies:  
C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Windows is a trademark of Microsoft Corporation.

Borland C++      Borland International, Incorporated.  
MicroFocus Cobol MicroFocus Corporation.

Other trademarks are trademarks of their respective companies.

-----

## Summary of Amendments

This is an interim update to the current edition of this book published by the ITSO. It is available in on-line form only.

This update contains corrections and additions to the original edition.

The following is a summary of the amendments included in this edition. Volume and Page number references refer to the original printed edition (SG24-4640-00).

### Interim update 0.1

Vol1 Page 31, 1.10 Exceptions  
    Change heading of exception table to say # instead of TRAP  
Vol1 Page 196, The Kernel Semaphore (KSEM)  
    Add an additional comment about Event KSEMs.  
Vol1 Page 338 ,Vol2 Page 274, Glossary  
    Re-worded glossary entry for BMP.  
Vol1 Page 350 ,Vol2 Page 287, Glossary  
    Added references to Vol 4 in VMKH entry.  
Vol1 Page 350 ,Vol2 Page 287, Glossary  
    Re-worded glossary entry for UVIRT.  
Vol2 Page 53, 2.2 Dump Decompression  
    DCOMP added to DF user guide.  
Vol2 Page 170, 3.4.5.6 Physical Device Driver Header  
    Corrected .D DEV note about DevInt  
Vol2 Page 198 .MA - Format Memory Arena Records  
    Added description of hashing table.  
Vol2 Page 227, 3.4.17.1 Idle PFs and 3.4.17.2 Free PFs

Clarify description of how to locate idle and free PFs.

Vol2 Page 251, 3.4.21 .PB - Display Blocked Thread Information  
Add an additional comment about Event KSEMs.

Vol2 Page 261, 3.4.24 Display User's Registers  
.R command EI flag should read DI flag when bit 9 is 0 for Flag register bit mnemonics.

Vol2 Page 266, 3.4.26 .S - Set or Display Default Thread Slot  
.S, command S option edited.

Vol4 Pages 27 and 32, 2.8.4 LogGetEntries  
Removed references to LogGetEntries. This API is not supported.

Vol4 Page 49, 3.2.2 System Anchor Segment (SAS)  
Correct offsets in SAS base section and inserted length field for SAS\_Info\_Data.

Vol4 Page 52 VMBH BMP Header Structure  
Structure added to miscellaneous section

Vol4 Pages 80, 82, 83, and 88  
Amended VMAR, VMAL, PF and VP structures to show correct layout of bit fields. Added pf\_block to PF structure.

Vol4 Pages 88  
Kernel Heap Structures added.

Vol4 Page 184, 3.7.3 System File Table Entry for OS/2 Warp 3.0  
Corrected SFT length of sft\_sfi and offsets of fields following.

Vol4 Page 210, 3.8.3 PDD IRQ Information Blocks  
Corrected title of PDD IQR to PDD IRQ

Vol4 Page 245, 4.4 Standard GDT Assignments  
Deleted note at end of table. Added new note before table.

Vol4 Page 280, 4.9 OS/2 FixPak to Build Level Cross-Reference.  
Corrected build levels for Warp GA, Connect and title of 8.200

Vol4 Page 210, 3.8.3 PDD IRQ Information Blocks  
Corrected VMOwner info for DIRQ and IRQI.

#### **Interim update 0.2**

Vol1 Page 101, Predefined Dynamic Trace Events  
MONCALLS added

Vol1 Page 103, TRCUST, The Dynamic Trace Customizer  
Change title

Vol1 page 130, DosEnterCriticalSection ....  
'crt' state is incorrectly described.

Vol1 page 134, The Dispatcher, Priorities and Dispatching Classes  
Added note at end of section on running in the kernel and device drivers.

Vol1 Page 135, The Status of a Thread  
'crt' state is corrected.

Vol1 Page 137, A Form for Unwinding Stacks  
Removed from INF version.

Vol1 page 202, Involuntary Suspension  
Critical Section: typo, helped should read held.

Vol1 page 202, Involuntary Suspension  
Pre-emption: added note about kernel mode.

Vol1 page 308, How to find the MQ of any Thread  
AAB at TLMA offset +0x8 (not +0xc)

Vol1 page 308, How to find the MQ of any Thread  
Expanded note at the bottom of the page.

Vol1 page 317, Finding the System Queue  
Current read pointer should be offset +0x0c (not +0x0e).

Vol2 Page 126, Q Command  
Typos: cases->causes, memu-> menu

Vol2 Page 129, 3.4.24 Display User's Registers  
R command NV flag value should be 0.

Vol2 Page 255, .PQ Scheduler Priority Queues  
.\_ptcbPriQRunner should say 'run' state

Vol2 Page 261, 3.4.24 Display User's Registers  
.R command NV flag value should be 0.

Vol3 Page 45, Major Code Assignments  
MONCALLS separately listed

Vol3 Page 218-245, PMSHAPI  
Obsolete tracepoints removed.

Vol4 Page 188, Record Lock Record for OS/2 Warp V3.0  
GDT\_FSC should read GDT\_RLR

Vol4 Page 88, Virtual Page Structure  
vpf\_s type should be S and vp\_flink type should be D.

#### **Interim update 0.3**

Vol1 Pages 245 14.1.2.1 Who Owns Virtual Memory and Who Allocated it?  
Correct typographic error: pseud-objects -> pseudo-object (on-line version only)

Vol2 Page 80, List of Internal Commands  
     .L should read L (printed edition only)

Vol2 Page 52 Dump Formatter Installation  
     In printed copy only, Note: DF&US.RET.EXE and DF&US.DEB.EXE should read df\_ret.exe and df\_deb.exe

Vol2 Page 214 3.4.16.2 Pseudo-Object Records  
     Correct typographic error in Notes: (psuedo)

Vol2 Pages 226 3.4.17.1 Free Page Frame Structures  
     Correct typographic error in Notes: (psuedo)

Vol2 Pages 227 3.4.17.2 Idle Page Frame Structures  
     Correct typographic error in Notes: (psuedo)

Vol3 Page 17 RETEP  
     Note added about modern C compilers.

Vol3 Page 5  
     Note added following first paragraph.

Vol3 Page 45 3.4 Trace Major and Minor Code Assignments  
     Table extended to include major codes reserved for other components.

Vol4 Pages 9 and 10, 2.1.2 DevHlp\_SysRAS  
     References to DevHlp\_SysTrace and DevHlp\_SysRAS should read DevHlp\_RAS.

Vol4 Page 14 2.2.1 Trace Buffer Structures  
     CHECK KEY: filed->field, Exclusively starts with Upper case E.

#### **Interim update 0.4**

Vol1 Page xx Acknowledgments  
     Add Joanna Hodgson.

Vol1 Page 9 1.3.5.2 Descriptor Flags  
     Correct definitions of bit 53 and 54

Vol1 Page 10 1.3.5.3 Descriptor Table Summary  
     Correct VDM considerations for IDT and LDT

Vol1 Page 196 The Kernel Semaphore  
     Clarify use of KSEM blockids

Vol1 Page 295 14.1.3.5  
     Added more useful PM message queue and window symbols

Vol1 Page 342 Interrupt Descriptor Table  
     Added information on VDM use of IDTs.

Vol1 Page 343 Local Descriptor Table  
     Added information on VDM use of LDTs.

Vol2 Page 14 Forcing a System Dump from the Kernel Debugger  
     Add .SYSDUMP command to first paragraph.

Vol2 Page 17-18 Forcing a System Dump from the Kernel Debugger  
     Add RegSA for Pentium processor support.

Vol2 Page 41 1.5 Kernel Debugger Breakpoints  
     Update XCPTBuildR3DispatcherStack

Vol2 Page 41 1.5 Kernel Debugger Breakpoints  
     Add DOS32R3EXCEPTIONDISPATCHER

Vol2 Page 45-46 1.6 Exception Logic  
     Correct information about local exception handlers used by the system and the exception handling logic. Add details of Dos32R3ExceptionDispatcher and Dos32ExceptionCallBack.

Vol2 Page 48 1.6.2 Exception Handling - Overview diagram  
     DosRaiseException flow corrected. Dos32ExceptionDispatcher

Vol2 Page 86-87 3.3.6 BL command  
     Update for I/O breakpoints.

Vol2 Page 89 3.8.8 BR  
     Add I/O breakpoint.

Vol2 Page 99 3.3.17.1 Descriptor formats Table 4  
     Clarify BIG C32 definitions

Vol2 Page 105 3.3.20.1 DP command  
     Add note about invalidity of PDEs under DF.

Vol2 Page 127 3.3.35.1 R command  
     Add note about addressing mode.

Vol2 Page 127 - 129 3.3.35.1 R command  
     Remove reference to 24-bit registers and move GDTB and IDTB to 32-bit registers.

Vol2 Page 128 3.3.35.1 R command  
     Add CR4 to 32-bit reg syntax diagram

Vol2 Page 131 3.3.35.1 R command  
     Add note on CR4 and correct 32-bit register

Vol2 Page 137 3.3.38 U command  
     Add note about V8086 mode addressing.

Vol2 Page 147 External Commands  
     .O command added.

Vol2 Page 147 External Commands  
     .SYSDUMP command added.

Vol2 Page 148 3.4.1 .? command



Updated note.

Vol2 Page 161 3.4.5.11 .D  
typo in warning formate->format

Vol2 Page 161 3.4.5 .D  
Note added about <512 byte segments

Vol2 Page 167 3.4.5.5 figure 33  
Add remark about fscrit

Vol2 Page 168 3.4.5.5 figure 33  
Add note about KSEM blockids

Vol2 Page 174 3.4.5.8 .D MFT  
Note added about ALLSTRICT kernel

Vol2 Page 196 .M  
Added note about defaults for FP29 and V4

Vol2 Page 182 3.4.5.11 .D SEM32 etc..  
pNname typo corrected no->not

Vol2 Page 190 3.4.1 .lm command  
Added I option. Updated Note. Update Results and Notes.

Vol2 Page 204 .MC  
Added DF icon

Vol2 Page 205 .MC  
Added additional note about early versions of DF

Vol2 Page 207 .MK  
Updated warning about freed memory.

Vol2 Page 210 .ML Results and Notes  
Updated note about the need for kernel symbols

Vol2 Page 213 .MO Results and Notes  
Added note about hmte use and interpretation.

Vol2 Page 225 .MP Syntax  
Added I option and clarified syntax.

Vol2 Page 226 .MP R and L option Warnings.  
Updated warnings for use of R and L

Vol2 Page 230 .MV R and L option Warnings.  
Updated warnings for use of R and L

Vol2 Page 237 .O command  
.O command added.

Vol2 Page 247 .PB Sta  
Note about fix

Vol2 Page 250 .PB Notes:  
Updated note on ChildWait.

Vol2 Page 258 3.3.35.1 .R command  
Add note about addressing mode.

Vol2 Page 266 .SYSDUMP  
.SYSDUMP command added.

Vol4 Page 41 and 89, 3.0 and 3.5 Scheduler control blocks  
ljmp structure added to list of scheduler structures.

Vol4 Page 96 3.5.1.7 Excepton Handling - Overview diagram  
DosRaiseException flow corrected. Dos32ExceptionHandler replaced \_xcptExceptionHandler.

Vol4 Page 125 long-jump buffer  
ljmp structure added

### **Interim update 0.5**

Introduction to the current edition  
Updated and fixed typos.

Vol1 Page 103 Chapter 8 TRCUST The Dynmaic Trace Customizer  
Remove duplicate chapter from volume 1

Vol1 Page 81 5.6.1.1 How to find the TSS  
Add additional fields to the TSS

Vol1 Page 129 11.1.2 Multiprocessor Methods - Spin Locks  
Correct formatting error in note about LOCK prefix.

Vol1 Page 142, 13.1.1.1 Address Space Arenas and Regions  
Add note about Protected Region post FP19 Warp 3.0

Vol1 Page 143, 13.1.1.1 Address Space Arenas and Regions  
Add note about Packed Region post FP19 Warp 3.0

Vol1 Page 255,256 Exploring Memory Management, Private Arena private data

Vol1 Page 260, Exploring Memory Managenent, Finding Who Owns Memory  
Note about .mam after 3.0 FP29 and 4.0 GA

Vol1 Page 322, 14.1.4.1 Ring 0 Loop Dump Analysis Example  
Add note about fsd and dd system owners after 3.0 FP29 and 4.0 GA

Vol2 Page 1, Kernel Debugger User Guide  
Added up to date URL and FTP information.

Vol2 Page 46, 1.6 Trap and Exceptions Processing  
Added information about VSU and clarified DelayHardError.

Vol2 Page 46, 1.6 Trap and Exceptions Processing  
Added exception handler return code values.

Vol2 Page 50, 1.6.4 Intercepting Exceptions and Traps  
Added information about VSU.

Vol2 Page 139, 3.3.39 Exception/Trap/Fault Vector Commands  
Added information about VSU.

Vol2 Page 140, 3.3.39 Exception/Trap/Fault Vector Commands  
Added information about VSU.

Vol2 Page 140, 3.3.39 Exception/Trap/Fault Vector Commands  
Added note about SMP.

Vol2 Page 217-219 .MO command  
Clarification of **own** and **hmte** fields. Remove 'main' from expression 'main executable' in the 'description' section.  
Add note about System Owner Objects on page 219 for 3.0 FP29 and 4.0 GA.

Vol2 Page 219 System Object Ids.  
Clarification of use of system object Ids.

Vol3 Page 5 TRCUST reference  
Update notes on restrictions.

Vol3 Page 13 2.2.2 TSF Header  
Add note about modname requirements

Vol3 Page 14 2.2.2 TSF Header  
Add node on major code range.

Vol3 Page 14 2.2.2 TSF Header  
Add node on maxdatalength default and range.

Vol3 Page 20 2.2.5.7 FMT Keyword  
Clarify use of %P

Vol3 Page 21 2.2.5.7 FMT Keyword  
Clarify use of %B

Vol3 Page 21 2.2.5.7 FMT Keyword  
Add %C formatting control

Vol3 Page 21 2.2.5.7 FMT Keyword  
Clarify use of %R

Vol3 Page 22 2.2.5.7 FMT Keyword  
Update use of %U and dump format

Vol3 Page 31 2.3 Formatting Trace Data  
Clarify use of %P and %R

Vol3 Page 45 3.4 Trace Event Major and Minor Code Assignment  
Add new major codes

Vol3 Page 163 Resource Manager Tracepoints  
New section

Vol3 Page 217 Multi-Media Extensions Tracepoints  
New section

Vol4 Page 70 Virtual Address Space Regions.  
Add diagrams for OS/2 V2.11 and OS/2 Warp V4.0

Vol4 Page 98 Thread Control Block for OS/2 Warp V4.0  
Add OS/2 Warp V4.0 tcb

Vol4 Page 125 Per-Task Data Area for OS/2 Warp V4.0 ALLSTRICT kernel  
Add OS/2 Warp V4.0 ALLSTRICT kernel ptdata

Vol4 Page 131 Per-Task Data Area for OS/2 Warp V4.0 RETAIL kernel  
Add OS/2 Warp V4.0 RETAIL kernel ptdata

Vol4 Page 167 Swappable Module Table Entry for OS/2 Warp V4.0  
Add OS/2 Warp V4.0 smte

Vol4 Page 117 3.7.1.5 Anonymous and Named Pipes  
Update diagram to show instances of named pipes.

Vol4 Page 199 3.7.10 Named Pipe Structures for OS/2 Warp V3.0  
Add pointer information about instances of named pipes.

Vol4 Page 237 System exceptions.  
Add note about XCPT\_PROCESS\_TERMINATE

Vol4 Page 243 4.3 Trap Screen Reference - System Internal Processing Error.  
Clarified line number info in an IPE.

Vol4 Page 244 4.3 Trap Screen Reference  
Correct typo formatter -> formatted.

Vol4 Page 244 4.3 Trap Screen Reference  
Added NMI Error Codes.

Vol4 Page 249 VM System Object Owner Ids.  
Clarification of use of system object Ids.

Vol4 Page 255 4.7 DevHlp Function Cross-Reference.  
Remove duplicate DevHlp\_DevDone

Vol4 Page 256 4.7 DevHlp Function Cross-Reference.  
Add new DevHlps for SMP and the Security feature.

Vol4 Page 280 4.8 Fix Pack to Build Level Cross-Reference.  
Updated for more recent fix packs and OS/2 Warp V4.0

#### Interim update 0.6

Vol1 Page 27 1.8.6 System Flags  
Add reference to the EFLAGS register.

Vol1 Page 99 7.2 TRACE and TRACE Processing  
Add note about dynamic tracepoints

Vol1 Page 210 13.1.2.3 Priority Inversion  
Reword last 5 paragraphs.

Vol1 Page 344 Glossary  
Updated PAI entry.

Vol2 Page 17-18 Forcing a System Dump from the Kernel Debugger  
Correct offset on RegSA for CR4.

Vol2 Page 89 3.8.8 BR  
Add note about I/O breakpoint address specification.

Vol2 Page 89 3.8.8 BR  
Add note about I/O breakpoint bug.

Vol2 Page 206 3.4.14 .MK Display Memory Lock Information Records  
Add note about HSTRICT kernel.

Vol2 Page 208 3.4.14 .MK Display Memory Lock Information Records  
Add note about the use of cs and eip for 16-bit callers.

Vol2 Page 231 3.4.18.1 Free Virtual Page Structures  
Correct typos in notes.

Vol2 Page 267 2.4.27 .T Dump the System Trace Buffer  
Add note about suspending the trace from the kernel debugger console.

Vol4 Pages 69 & 80 Memory Management Control Block Reference  
Added Physical Arena Information structures.

Vol4 Page 98 Thread Control Block for OS/2 Warp V4.0  
OS/2 Warp V4.0 tcb was not added correct by #0.5

Vol4 Page Page 238 4.2 OS/2 System Exception Codes  
Add XCPT\_UNKNOWN\_ACCESS to P2 = -1 for XCPT\_ACCESS\_VIOLATION.

#### **Interim update 0.7**

Vol1 page 127 10.2 Steps to Diagnose a Loop  
Typo in online version: theas-> threads

Vol1 page 129 11.1.2 Multiprocessor Methods - Spin Locks  
Typo in online version: .Exchange -> Exchange

Vol1 page 130 11.1.3 DosEnterCriticalSection and DosExitCriticalSection  
Reword second sentence for better understanding.

Vol1 page 147 13.1.1.2 Virtual Address Space Management  
Add \_ahvmhShr and correct label typos for \_ahvmShr and \_ahvmSys

Vol2 Page 45-46 1.6 Exception Logic  
Add parameter information for **\_XCPTBuildR3DispatcherStack**.

Vol2 Page 190s and 192 3.4.10 .LM - Format Loader Structures  
Add references to RASKDATA.

Vol2 Page 206 3.4.14 .MK - Display Memory Lock Information Records  
Add references to RASKDATA.

Vol2 Page 247 .PB BlockId  
Clarify DosSem

Vol3 page 5 TRCUST Reference  
Added information on latest system trace tools.

Vol3 page 5 TRCUST Reference  
Updated note on dynamic trace restrictions

Vol3 page 7-8 1.2.1 Invoking the Trace Customizer  
Updated syntax diagram and description for TRCUST 3.06

Added /D, /L, /NODE, /NOLN, /RM, /RS, /I, /P, /PREINV, /RAS

Vol3 page 8 2.1.3.1 Source Level Symbolic Support  
Updated information on supported compilers and added reference to VisulAge and DEDEL.

Vol3 page 9 2.1.3.2 MAP File Support  
Added reference to /I

Vol3 page 9 2.1.3.3 Building a Module  
Added reference to DEBDEL

Vol3 page 11 2.1.2 TDF and TFF File Usage  
Added references to latest tools (DTRACE, TRACEGET, TRSPOOL, TRACE /Q)

Vol3 page 13 2.2.2 TSF Header  
Added TDFID and changed MAJOR to be optional.

Vol3 page 13 2.2.2 TSF Header  
Updated major code range for RAS Enhancements

Vol3 page 14 2.2.3 Typelist Definition  
Clarified use of TYPELIST

Vol3 page 14 2.2.4 GroupList Definition  
Clarified use of GROUPLIST

Vol3 page 15 2.2.5 Tracepoint Definition

Added new RETEP options to syntax.

Vol3 page 16 2.2.5.1 MINOR Keyword  
Added reference to /D

Vol3 page 16 2.2.5.2 TP Keyword  
Clarified use of @STATIC. Referenced MAKETSF. Referenced /PREINV. Updated RETEP description for new sub-keywords. Corrected invalid opcode list.

Vol3 pages 20-22 2.2.5.7 FMT Keyword  
Clarified use of %P, %C, %U. Added note on TRACEFMT dump format. Clarified not for CMVC users.

Vol3 page 28 2.2.5.15 Address Specification  
Added note about /I and name length limitations.

Vol3 page 28 2.2.5.15 Address Specification - Flat Register Form  
Updated 32-bit register lists.

Vol3 page 29 2.2.5.15 Address Specification - Segmented Register Form  
Updated 16-bit register lists. Clarify use of R prefix.

Vol3 page 31 2.3 Formatting Trace Data  
Clarify use of %P and %R. Add details of prefix format.

Vol3 pages 38 2.5.1 External Messages  
Remove this section. These messages are not generated.

Vol3 pages 39-42 2.5.2 Internal Messages  
Rename section and add new messages for TRCUST 3.06

Vol3 page 44 3.2 Group Qualifiers  
Added new groups.

Vol3 page 44 3.3a DosXxxx API Pre-invocation Tracepoints  
Added section describing return addresses.

Vol3 page 45 3.4 Major Code Cross-Reference  
Added new assignments.

Vol3 page 55 3.4.7 Kernel Services Trace Events  
Add indirected APIs

Vol3 page 96 3.4.9 Kernel Services Trace Events  
Add tracepoints for the following APIs:  
 Dos32CancelLockRequest  
 Dos32SetFileLocks  
 Dos32ProtectSetFileLocks  
 DosCreateSpinLock  
 DosAcquireSpinLock  
 DosReleaseSpinLock  
 DosFreeSpinLock  
 Dos32IProtectRead  
 Dos32IProtectWrite

Correct the following tracepoints:  
 Dos32PMPostEventSem  
 Dos32PMWaitMuxWaitSem  
 Dos32PMWaitEventSem

Vol3 page 97 3.4.7 OS2KRNL Trace Events  
Remove this section since identical with OS2KRNL

Vol3 page 164 3.4.21 DOSCALL1.DLL Trace Events  
Add indirected APIs

Vol3 page 168 3.4.21 DOSCALL1.DLL Trace Events  
Added parameters to Dos32RaiseException, DosFsSemClear and DosFsSemRequest.

Vol3 page 168 and 170 3.4.21 DOSCALL1.DLL Trace Events  
Added Dos32R3ExceptionDispatcher, Dos32ExceptionCallBack, xcptExecuteExceptionHandler, UT16\_RETURN and UT32\_RETURN tracepoints.

Vol3 page 209 3.4.30 QUECALLS.DLL Trace Events  
Add indirected APIs

Vol3 page 209 3.4.33 SESMGR.DLL Trace Events  
Add indirected APIs

Vol3 page 203 3.4.28 OS2CHAR.DLL Trace Events  
Added CharBuffer to KbdStringIn Post-Invocation.

Vol3 page 345 3.4.51 PMGPIR Trace Events  
Remove this section since identical with PMGPI

Vol3 page 345 3.4.51 PMGPIR Trace Events  
Remove this section since identical with PMGPI

Vol3 page 382 3.4.54 PMGPID Trace Events  
Remove this section since identical with PMGPI

Vol4 page 1 Chapter 1 CONFIG.SYS RAS Commands  
Change title to CONFIG.SYS RAS Statements

Vol4 page 1 Chapter 1 CONFIG.SYS RAS Commands  
Added SCKILLFEATUREENABLED.

Vol4 page 1 Chapter 1 CONFIG.SYS RAS Commands  
Added RASKDATA

Vol4 page 2 1.2a RASKDATA  
Inserted RASKDATA section.

Vol4 page 2 1.3a SCKILLFEATUREENABLED  
 Inserted SCKILLFEATUREENABLED.

Vol4 page 4 1.6 SUPPRESSPOPUIS  
 Updated for SUPPRESSPOPUIS=0

Vol4 page 6 1.10 TRAPDUMP  
 Added PD parameter and Ctrl-Alt-F10-F10 key sequence.

Vol4 page 6 1.10 TRAPDUMP  
 Added reference to the TRAPDUMP command.

Vol4 page 7 1a Miscellaneous RAS Command Command Utilities  
 Added new chapter to describe TRAPLOG, TRAPDUMP and SYSDUMP

Vol4 Page 52 VMBH BMP Header Structure  
 Structure added more examples of BMP use.

Vol4 page 69 & 80 Memory Management Control Block Reference  
 Added Per Arena Page Table Data (PGDATA)

Vol4 Page 88 VMKSH Structures  
 Fix swappable heap header offsets.

Vol4 page 172 3.7 File System Block Reference  
 Correct title

Vol4 page 242 4.3 Trap Screen Reference  
 Correct wording of xSLIM descripton.

Vol4 Page 255 4.7 DevHlp Function Cross-Reference.  
 Correct typo in DevHlp\_RegisterDeviceClass

Vol4 Page 255 4.7 DevHlp Function Corss-Reference.  
 Added: helper functions 0x7f - 0x83

Vol4 page 257 4.7a Device Driver Strategy Commands  
 Added table to system cross-reference.

---

## Preface

Debugging problems is essentially an iterative process of hypothesis, test and conclusion that aims to eliminate the irrelevant and therefore focus on the probable causal area.

To engage this process successfully one needs to be equipped with an innate ability to think laterally coupled with sufficient knowledge of the environment in which the problem persists and above all else to be able to use the tools that extract information from the system under diagnosis.

This scenario applies as much to first level problem determination (PD) as it does to the software developer who is engaged in detailed analysis of his programs' behaviour.

Information and tools to aid first level problem determination is relatively accessible. Technical literature is available from IBM and books stores that will fulfil the needs of the first level PD analyst. For example, the reader is invited to consult the following IBM Red-book publications to achieve an all-round high-level technical appreciation of the OS/2 environment:

The Technical Compendium Volume 1 - Control Program

The Technical Compendium Volume 2 - DOS and Windows Environment

The Technical Compendium Volume 3 - Presentation Manager and Workplace Shell

The Technical Compendium Volume 4 - Application Development

The Technical Compendium Volume 5 - The Print Sub-system

The problem analysis level that is less well provided for is that which involves internal knowledge of the OS/2 operating system and its diagnostic tools. This is the level at which Service personnel, System Programmers and Software Developers work. It is this audience to which the OS/2 Debugging Handbooks are directed.

An inevitable consequence of working at a deep technical level is that the amount of information one could amass is vast. Given time constraints and the need to publish useable material before it became obsolete we had to make certain compromises for the first edition. The following principles guided us in making decisions about which material to include:

Material that is adequately documented elsewhere is referenced but not included.

Accurate reference documentation for the diagnostic tools and facilities available for OS/2 has been given priority over worked examples and OS/2 Internals reference material.

Internals information has centred around the base operating system - that is, the kernel.

We hope to remedy some of these short-comings in future revisions of this book and in companion volumes. Updates to the on-line version of this book will be made available via the Developer Connection CDROM.

The current printed edition contains full reference material for the following OS/2 System diagnostic facilities:

System Trace

System Dump

Kernel Debugger

In addition to these topics we have included an introductory guide to problem determination. This provides a resumé of the hardware and software environment and an introduction to using the dump formatter and kernel debugger.

Throughout this book we assume the availability of and familiarity with two co-requisite publications:

The Intel Pentium Family User's Manual, Volume 3: Architecture and programming manual, ISBN 1-55512-227-2, Intel order number 241430-003.

This should be consulted as the authoritative source for hardware architectural information.

The Design of OS/2 by H.M. Deitel and M.S. Kogan.

This should be consulted for an overview of the internal operation and architecture of OS/2.

This book is supplied with a CDROM whose contents are:

- Sample exercises to accompany [Volume 1, Introduction to Debugging](#). These take the form of system dumps of typical problems in application programs.
- On-line version of this book. This is slightly more up-to-date than the printed version and includes more worked examples. This is an **.INF** file and should be viewed using the OS/2 **VIEW.EXE** program. Much use has been made of **hypertext** links, which direct the user to the glossary. From the glossary it is possible to link to related material in other sections of the book.
- The OS/2 Problem Determination Package (OS2PDP), which includes the dump formatter, symbol files, and trace customiser (**TRCUST**).

Unless otherwise stated the material in this book may be assumed to be applicable to OS/2 Warp version 3.0 (ALLSTRICT Kernel).

As indicated above, work on this subject matter can never be complete. We intend build on and update the material in this edition. In order to address the areas in most need of attention we invite the reader to fill in the Reader's Comment Form with their suggestions.

-----

## Related Publications

Throughout this book we assume the availability and familiarity with two co-requisite publications:

The Intel Pentium Family User's Manual, Volume 3: Architecture and Programming Manual, ISBN 1-55512-227-2

The Design of OS/2 by H.M. Deitel and M.S. Kogan, ISBN 0-201-54889-5

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

INTEL486 Microprocessor Family Programmer's Reference, published by the INTEL Corporation. ISBN 1-55512-159-4 Intel order No. 240486-002

The Intel Pentium Family User's Manual, Volume 3: Architecture and programming manual, published by the INTEL Corporation. ISBN 1-55512-227-2, Intel order number 241430-003.

The Design of OS/2 by H.M. Deitel and M.S. Kogan, published by Addison Wesley. ISBN 0-201-54889-5

The OS/2 Technical Library Control Program Programming Reference Version 2.00., S10G-6263-00.

The Design Workbook for Cruiser (OS/2 2.0) Volumes 1 - 13. IBM Internal Publication

The Final Programming Functional Specifications for Cruiser (OS/2.0). IBM Internal Publication.

Installable File Systems for OS/2 Version 2.0. IBM OEMI Publication.

The OS/2 1.0 System Trace Facility. IBM OEMI publication.

The PS/2 2.0 Error Logging Functions. IBM OEMI publication.

OS/2 2.0 Proc Lang 2/REXX Reference, S10G-6268-00

OS/2 2.0 Proc Lang 2/REXX User Guide, S10G-6269-00

OS/2 WARP Control Program Programming Guide, G25H-7101-00

OS/2 WARP Control Program Programming Reference, G25H-7102-00

OS/2 WARP PM Basic Programming Guide, G25H-7103-00

OS/2 WARP PM Advanced Programming Guide, G25H-7104-00

OS/2 WARP GPI Programming Guide, G25H-7106-00

OS/2 WARP GPI Programming Reference, G25H-7107-00

OS/2 WARP Workplace Shell Programming Guide, G25H-7108-00

OS/2 WARP Workplace Shell Programming Reference, G25H-7109-00

OS/2 WARP IPF Programming Guide, G25H-7110-00

OS/2 WARP Tools Reference, G25H-7111-00

OS/2 WARP Multimedia App Programming Guide, G25H-7112-00

OS/2 WARP Multimedia Subsystem Programming, G25H-7113-00

OS/2 WARP Multimedia Programming Reference, G25H-7114-00

OS/2 WARP PM Programming Reference Volume I, G25H-7190-00

OS/2 WARP PM Programming Reference Volume II, G25H-7191-00

Technical Reference - Personal Computer AT, Part Number 1502494

Personal System/2 and Personal Computer BIOS Interface Technical Reference - Part Number 68X2341

-----

## International Technical Support Organization Publications

OS/2 Warp Connect, GG24-4505

OS/2 Warp Generation, Vol.1, SG24-4552

OS/2 Warp Version 3 and BonusPak, GG24-4426

Multimedia in Warp, GG24-2516

The Technical Compendium Volume 1 - Control Program, GG24-3730

The Technical Compendium Volume 2 - DOS and Windows Environment, GG24-3731

The Technical Compendium Volume 3 - Presentation Manager and Workplace Shell, GG24-3732

The Technical Compendium Volume 4 - Application Development, GG24-3774

The Technical Compendium Volume 5 - The Print Sub-system, GG24-3775

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get a catalog of ITSO redbooks, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

A listing of all redbooks, sorted by category, may also be found on MKTTOOLS as ITSOCAT TXT. This package is updated monthly.

How to Order ITSO Redbooks  
IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER.  
Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-445-9269. Most major credit cards are accepted. Outside the USA, customers should contact their local IBM office. Guidance may be obtained by sending a PROFS note to BOOKSHOP at DKIBMVM1 or E-mail to [bookshop@dk.ibm.com](mailto:bookshop@dk.ibm.com).

Customers may order hardcopy ITSO books individually or in customized sets, called BOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

---

## ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser (such as WebExplorer from the OS/2 3.0 Warp BonusPak) to the following URL:

<http://www.redbooks.ibm.com/redbooks>

IBM employees may access LIST3820s of redbooks as well. Point your web browser to the IBM Redbooks home page:

<http://w3.itsc.pok.ibm.com/redbooks/redbooks.html>

---

## Acknowledgements

The authors of this book are:

Pete Guy  
IBM SDO, Austin

Richard Moore  
IBM PSP EMEA

Redbook project developed by:

Tim Sennitt  
ITSO Boca Raton, Center

This book could not have reached publication without the encouragement, help and support from a number of colleagues and friends. In particular we would like to thank the following:

Tim Sennitt for his help in preparing the printed material and doing much of the donkey-work to bring this to publication.

Joanne Rearnkham, Barry Bryan and David Jaramillo for their support in enabling access to the materials necessary to produce this book.



Chris Perritt and Glen Brew for making available the original Design Workbook and Functional Specifications for OS/2 2.0.

Charlie Schmitt for his original work on converting the kernel debugger code into a dump formatter.

Jeff Mielke and David Jaramillo for their work on PMDF, the structure compiler and continued work on the dump formatter.

Allen Gilbert for making available documentation on System Trace, which has been reproduced in an edited form in this book. Also, for making available an early version of the dump formatter without which it would not have been possible to develop the original Dump Formatter class.

Doug Azzarito for supplying the material on Kernel Debugger Remote Debug Set-up.

James Taylor for providing the basis of the lab exercises relating to PM hangs.

Marie Jazynka, one of the first OS/2 debuggers, for patient encouragement of a great many OS/2 debugging people.

Joanna Hodgson for preparing some of the Warp 3.0 fix pack 29 and Warp 4.0 updates.

Our management teams, without whose foresight and support none of this work would ever have started. These include:

- Hermann Lamberti General Manager for PSM EMEA; Gordon Bell - director PSM EMEA Technical Marketing; Chris Brown - manager PSM OEM and Enterprise Technical Marketing and Brian Rose - manager PSM Project Office; Roy Aho - Director of the Solution Developer Technical Support Center, for encouraging the beginnings of this several years ago; Terry Gray, manager of Platform Competency and Operation, within Solution Developer Technical Support, Austin.

Finally to Sarah-Jane for supporting many very extended working days and weeks.

-----

## ITSO Technical Bulletin Evaluation

### The OS/2 Debugging Handbook Library

**SG24-4640-00**

Your feedback is very important to help us maintain the quality of ITSO Bulletins.

**Please either print out this questionnaire and complete it or complete it online. Then return it using one of the following methods:**

- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to [REDBOOK@VNET.IBM.COM](mailto:REDBOOK@VNET.IBM.COM)

**Please rate on a scale of 1 to 5 the subjects below.**

**(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)**

Overall Satisfaction	_____		
Organization of the book	_____	Grammar/punctuation/spelling	_____
Accuracy of the information	_____	Ease of reading and understanding	_____
Relevance of the information	_____	Ease of finding information	_____
Level of technical detail	_____	Completeness of the information	_____
Value of illustrations	_____	INF quality	_____

**Please answer the following questions:**

a) If you are an employee of IBM or its subsidiaries: Do you provide billable services for 20% or more of your time?	Yes_____	No_____
Are you in a Services Organization?	Yes_____	No_____

b) Are you working in the USA? Yes\_\_\_\_ No\_\_\_\_  
c) Was the Bulletin published in time for your needs? Yes\_\_\_\_ No\_\_\_\_  
d) Did this Bulletin meet your needs? Yes\_\_\_\_ No\_\_\_\_  
If no, please explain:

---

---

What other topics would you like to see in this Bulletin?

---

---

What other Technical Bulletins would you like to see published?

---

**Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK! )**

Name

---

Company or Organization

---

Address

---

---

Phone No

---

E-Mail

---

-----

## Introduction to the current edition

Recent releases of OS/2 have introduced an number of so called ***RAS enhancements*** and fixes to the System Debugging facilities. These enhancements have been released mainly in two major fix packs:

Fix Pack 29 for Warp 3.0 and base Warp 4.0

Fix Pack 35 for Warp 3.0 and Fix Pack 10 for Warp 4.0

All RAS enhancements are available with the base release of OS/2 Warp E-Server

This edition of the The OS/2 Debugging Handbook has been updated to include information on some of these new debugging facilities. For interim update 0.5 I have concentrated on updating the system structures and trace information for OS/2 Warp V4.0. Notes have been added throughout where the internal implementation differs from earlier releases.

Interim update 0.7 contains major updates to TRCUST, RASKDATA, SYSDUMP, TRAPDUMP, TRAPLOG and the System Trace Reference that reflect the latest RAS enhancements. It also includes additional information on the SYS317x exception popup message.

Other facilities that were introduced with these releases of OS/2, but are not yet covered, include:

- DTRACE - the low-level dynamic trace utility

- System trace enhancements

- Software trace

- PROC\_DUMP, PDUMPSYS and PDUMPUSR utilities.

- PMDf enhancements

- TRACEFMT enhancements

- FFST/2

- Error Logging V2

- New REXX EXECs

- System Anchor Block

Future updates to the The OS/2 Debugging Handbook will include information on these topics together with updates on Warp SMP.

For further information on these new facilities the reader is referred to the **README.DBG** file, which is distributed with all fix packs that include the debugging enhancements.

---

## Approach to Problem Solving

In order to succeed at low-level program problem diagnosis, one must have several skills. None of these is particularly difficult, but many are foreign to today's programmers.

At first, it will appear that each problem is solved with a different technique. Study of the methods used to solve problems yields the fact that the several skills are used as appropriate, virtually as subroutines, and without thought, by experienced analysts.

The intent of this material is to provide the basic knowledge and to illustrate each of the skills separately, to aid understanding. Trying to solve problems without the basic skills can be extremely frustrating, at best.

The fundamentals include knowledge of hardware operation, software conventions, and basic use of tools to display the data sought. Once the fundamentals are understood, it is time to begin using them to solve problems, because one can then concentrate on building the problem solving skill.

Application traps are perhaps the easiest problems to approach, so they are explained after the basic skills. Similarly, traps in privileged code are only incrementally more difficult.

Once some experience in solving traps has been gained, it is reasonable to extend one's skills by exploring reasons for waits and loops, collectively known as hangs, or to learn the additional functions provided by Symmetric Multi-Processor (SMP) systems, as well as the challenges in properly serializing them when needed.

---

## List of Necessary Skills

The fundamental skills are:

- A good knowledge of how the hardware protection mechanisms work.

A good knowledge of what any instruction actually does.

A good knowledge of a few primary software conventions:

How a stack is used and what information is in it.

How to use the stack data for debugging.

How to use optional program documentation to get from a failing instruction to the actual line of the program which contains it.

How to find the program's variables in storage.

How to obtain the above documentation for some IBM languages.

How to collect a dump of a system at the point of failure.

How to use the available analysis tools.

How to determine the owner of a part of storage,  
and which processes have access to that storage.

And that's what this material is designed to teach!

---

## Collecting Documentation

If the problem can be reliably reproduced in a development environment, do it. This is the fastest way to get the problem fixed. When you cannot, try to get a good set of starting documentation.

It is possible to acquire and install a replacement for the OS/2 kernel which is the same as the one being replaced, except that it has debugging facilities and a debug interface to a serial port, COM2. If you install the wrong debug kernel, no one can predict the results. If you install the correct version, you will need to have a terminal emulation program (or ASCII terminal) to access the debug interface. The capabilities of this debug tool are essentially unlimited, and there is no protection from accidental entry errors. Its use is not a trivial task, nor one to be lightly undertaken.

It is often possible to collect enough information about a problem to diagnose its cause by creating customized trace entries specifically for that particular problem. For this to work well, the problem must be reproducible, and the trace buffer must be captured while the data gathered is still present.

Most people who have worked in a technical support role will agree that often the largest obstacle to solving a problem is collecting enough useful information about it. We will briefly discuss how to get enough useful data that problem solving can start in most cases. Be aware that frequently there will be some additional useful information, which can be gathered when the need for it is discovered, and that what is outlined here is not a complete list, by any means.

It is important to collect as complete a set of volatile data as possible from a single failure. If it is not gathered, it will be lost, perhaps requiring another occurrence of the problem in order to get needed information.

It is generally possible to use either an interactive debugger or a dump to diagnose either traps or hangs in an application.

For application problems, particularly traps, a good set of documentation includes the following:

- A statement of what sequence of events leads to the problem
- The trap screen, if a trap is involved
- A storage dump, with system trace data
- All the executable modules involved in the failure
- Optional application documentation, including:
  - all source files
  - .map files, produced by the linker
  - .lst and .cod or .asm files, produced by the compiler

The storage dump is the only thing which is volatile. The rest can be collected whenever the need is discovered. To collect the first item, perform the following steps:

**Note:** THIS WILL DRASTICALLY CHANGE OS/2 BEHAVIOUR WHEN A TRAP OCCURS. OS/2 WILL not CONTROL THE FAILURE, BUT WILL INSTANTLY AND IRREVOKABLY STOP THE SYSTEM, AND INITIATE A STORAGE DUMP. THERE WILL BE NO SHUTDOWN OF THE WORKPLACE SHELL, DATABASES, FILE SYSTEMS, (or lazy-write buffers,) OR ANYTHING ELSE. IT CAN BE AS DISRUPTIVE AS A POWER FAILURE. IT IS POSSIBLE TO LOSE FILES, OR PARTS OF FILES, but unlikely.

Prior to WARP: execute the command `CREATEDD A:` This will prepare a diskette for taking a dump. The diskette will work only once. This is not required for WARP, nor for later levels of 2.11. A quick way to discover if it is required is to read the prompt which asks for the diskette at the beginning of the process. If `CREATEDD` is required, the prompt asks for the diskette prepared by `CREATEDD`, otherwise it asks for a formatted diskette.

Preparation:

1. Save the current `CONFIG.SYS`
2. Edit `CONFIG.SYS`
  - a. If the line is not already present, add a line which reads `TRAPDUMP=ON`
  - b. add a line which reads `TRACEBUF=63` to enable the system trace
  - c. add a line which reads `TRACE=ON` to turn on the system trace
  - d. optionally, add a line which reads `TRACE=OFF,4,6,7`
  - e. optionally, turn `LAZYWRITE` off, so data goes directly to disk.
3. Locate some formatted diskettes to use for a storage dump.

Estimate about 2 Megabytes of RAM per diskette; usually one diskette more than that number is needed. For very large systems, estimate 1.5 meg per diskette. The dump process WILL NOT format.

4. Reboot the system so that the changes take effect.
5. Restore the original `CONFIG.SYS`, so you do not have to reboot an extra time to put things back to normal, after collecting the dump.

Acquiring the storage dump:

1. Cause the application to trap, that is, reproduce the problem.
2. Insert the `CREATEDD` diskette, if created, otherwise insert the first formatted diskette.
3. If you can read the screen, follow directions every time you hear one or more beeps.
4. If you cannot read the screen, you can still successfully get a dump, by listening for a beep. Insert the next diskette every time you hear a single short beep. When the dump is almost complete, there will be a very distinctively different series of beeps. At this point, reinsert the first diskette.
5. Very soon after the first diskette is reinserted, the dump will complete. Remove the diskette.
6. OS/2 will reboot automatically in most cases. Expect autocheck to run on HPFS drives during the boot.
7. Run `CHKDSK` on the drives as soon as convenient.

-----

## Hardware Architecture

This section explains how the hardware operates in protected mode, what forms of protection exist, how they operate, and what happens when a program attempts to violate one or more of the protection mechanisms.

The three protection mechanisms in 32-bit OS/2 are:

1. Privilege

2. Description
3. Address mapping

**Note:** All three are active at all times when 32-bit OS/2 is running protected mode programs. Only address mapping is active when 32-bit OS/2 is running a VDM in V86 mode.

---

## Address Components

All addresses in x86 processors are composed of two parts:

**Note:** Addresses are usually written with a colon separating the two parts, for example, selector:offset.

1. A segment or selector
2. An offset

The offset part will be covered during the review of typical machine instructions, because it is straightforward, and the same in real and protected modes.

**Note:** These two parts are implicitly or explicitly specified by every instruction that references memory for either or both operands. Generally, the selector is implied and the offset is specified but there are exceptions to this.

---

## NEAR & FAR Addresses

Because there are two parts of an address and an item may or may not be in a current segment, there are two ways to specify the address of a data item.

A NEAR ADDRESS is an offset without specifying a selector. This is a very efficient way to address data because the overhead of loading a selector register and fetching the descriptor is avoided. The selector to use is implied, and is normally already loaded.

A FAR ADDRESS contains both a selector and an offset in protect mode. This is slower and more cumbersome because both address components must be specified as well as causing the overhead of altering a selector register. When a far address is displayed from storage (as two words), the offset will be seen in the left word, and the segment or selector in the right word.

A FAR ADDRESS contains a segment and an offset in real or V86 mode. The overhead is not so bad as in protect mode, because there are no descriptors to fetch when a segment register is loaded.

---

## Real Mode and V86 Mode

Real and V86 Modes

CS = Code Segment	SS = Stack Segment
DS = Data Segment	ES = Extra Segment

for 386 and later,

FS = another data segment	GS = another data segment
---------------------------	---------------------------

In REAL or V86 modes, say 'segment registers'  
In PROTECT MODE, say 'selector registers'

**Note:** In real mode each segment register has a 16-bit number. The segment number is shifted left 4 bits, then added to the offset value. There is no checking of any kind.

```
DS=1234,    offset=5678

           12340
           5678
           ----
          179B8
```

**Note:** This is equivalent to any of the following:

```
segment 179B, offset 8;

segment 1790, offset B8;

segment 1267, offset 5348;

or many other possibilities.
```

-----

## Protected Mode

In protected mode, all storage is described by the hardware, using tables maintained by the software. The description includes the location, and size of the storage segment, as well as the type of storage. The storage type further constrains how it may be used.

This section concentrates on the selector part of the address because the offset is handled in a very simple and consistent fashion once the memory segment has been located and the validity of the access has been verified.

-----

## Descriptors

A selector specifies a descriptor, which describes a memory segment. The attributes described include the base or starting address of the memory segment, the size of the segment and what accesses are allowed.

Protected mode addressing in a 386 or later begins with Descriptor Tables which are described by hardware registers. There are three Descriptor Tables, each of which is discussed below after supplying the format of individual descriptors. The tables contain the descriptors and the descriptors are selected by an interrupt number or by the content of a selector register.

An application descriptor is required for all accesses to instructions and to data. For most segments, the limit is the largest valid offset. If the offset is larger than the limit, a general protection exception occurs. The exception to this rule occurs for data segments which are 'expand down'. In this case, the offset must be greater than the content of the limit field. The system stack ( ring 0 ) is an example of an Expand Down segment.

To find the linear address of the data element, the processor adds the offset ( obtained from the instruction ) to the base address of the segment. That's the end of the discussion for offsets!

There are three distinct kinds of data recognized by the processor:

- Stack, which holds temporary data, parameters and return addresses
- Code, which is instructions for the processor to execute
- Data, which is used to hold data which is available for longer than the lifetime of any one function or routine.

The primary distinction between stack and data is that data segments begin at offset zero and expand upward ( to the limit ) while stack segments begin at the highest offset and expand downward ( to just greater than the limit ). Many language implementations use data segments for their stack, which is perfectly acceptable, but it makes it impossible to 'grow' the stack.

The descriptor for a memory reference is found by using the appropriate selector as the index to a table or, if you ignore the 3 lower bits, as an offset to the table, since descriptors are 8 bytes long.

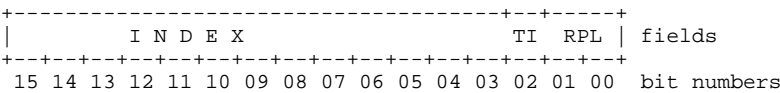
---

# Selector Format

In protect mode, a Selector has 3 fields:

- 1. Index, the left 13 bits, bits numbered 15-3  
This is an index into a descriptor table
- 2. Table indicator, one bit, bit number 2  
0 means GDT  
1 means LDT
- 3. RPL, the right 2 bits, numbered 1 & 0.  
Requested Privilege Level.  
perceived as a two bit value, range 0 to 3 00=most privileged, or ring 0; 11=least privileged, or ring 3.

**Note:** The position of the bits makes a selector (with its 3 low order bits turned off) the offset into the table.



# Privilege Levels

The point of privilege levels is to prevent a program from accessing a storage object that is more privileged than the program itself. Generally, this means that application programs are not able to access storage used by supervisory programs in any way. This also means it is safe to keep descriptions of storage used by the system in a descriptor table that can be accessed by applications, because the application cannot use those descriptors.

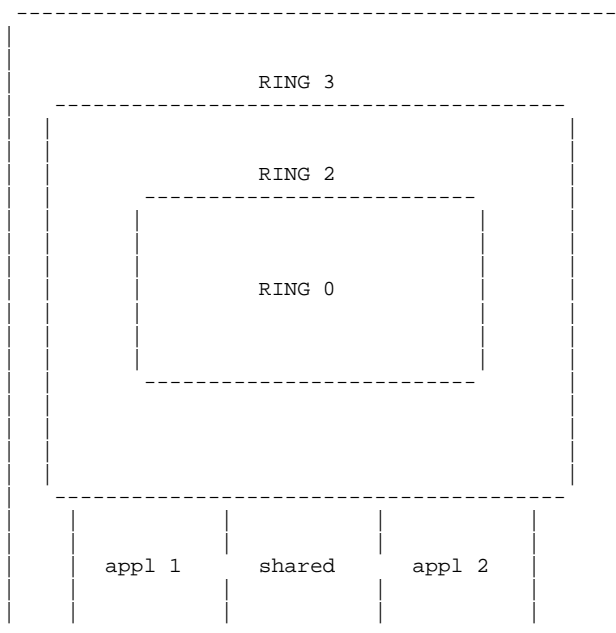
There are actually 3 distinct privilege levels associated with every storage access, and testing privilege level is a two-step process. The privilege level used to access a storage operand is the less privileged of CPL and RPL. The first step is to determine the actual privilege level with which to attempt the access. The second step is to compare the privilege level of the storage object (from the descriptor) to the result of the first step.

DPL	Descriptor Privilege Level.	Bits 45 & 46 of descriptor.
RPL	Requested Privilege Level.	2 low order bits of selector.
CPL	Current Privilege Level.	2 low order bits of CS.

A more privileged (lower numbered) program may access the storage objects of a less privileged program. This is how the operating system returns structures and fills in data areas for an application.

Any attempt by a less privileged (higher numbered) program to access in any way a storage object which is more privileged generates a general protection exception.





-----

## Descriptor Tables

There are three tables which hold descriptors.

The three tables are:

1. The Global Descriptor Table or GDT, describes memory objects which are accessible to all processes.

The GDT is located by means of a hardware register called the GDTR which contains the linear address and length of the GDT.

2. The Local Descriptor Table or LDT, describes memory objects which are unique to one process or are shared among a few processes by design.

The LDT is located by means of a hardware register called the LDTR which contains a selector. The descriptor referenced by this selector must be a system descriptor which describes an LDT.

3. The Interrupt Descriptor Table or IDT, has gates that specify interrupt handler entry points.

The IDT is located by means of a hardware register called the IDTR which contains the linear address and length of the IDT. The interrupt number is used to index into this table when an interrupt occurs.

-----

## Descriptor Fields

Type	Tells what kind of object is described
Application types:	Code, Data
System types:	LDT, TSS, Call Gate, Irpt Gate

Base	Linear address of object
Limit	Defines the size of a storage object
DPL	Privilege level defines which ring(s) can access the described object

```

+-----+-----+-----+-----+-----+-----+-----+-----+
LIMIT 00-15  BASE 0-23  TYPE S DPL P LIMIT 16-19  FLAGS  BASE 24-31
+-----+-----+-----+-----+-----+-----+-----+-----+
  0         1         2         3         4         5         6         7
                                byte offsets

```

Display a descriptor with 'DB' to see it in this form.

Notes:

TYPE is what kind of object is described  
 S is descriptor category; 0=system, 1=code or data  
 PL is privilege level of object described  
 P is the present bit; 1=present, 0=not present

## Descriptor Flags

Bit 55 Granularity: (G) 0=limit is in bytes, 1=limit is in 4K pages

Bit 54 Default address and operand (D/B) size: 0=16 bit, 1=32 bit

### Note:

In code segment this bit (called the D bit) governs the default address and operand size.

In a data segment this bit is ignored.

In a stack segment this bit (called the B bit) determines whether SP (B=0) or ESP (B=1) is used by instructions that implicitly reference the stack, for example PUSH, POP, CALL and RET.

Bit 53 Reserved, must be zero.

Bit 52 Unused by hardware, used by OS/2 to indicate UVirt

Bit 47 Present: (P) 1=segment is present, 0=segment is not present

Bits 46 & 45 Privilege Level: 00=most, 11=least

Bit 44 Segment type: 0=system segment, 1=application segment

Bit 40 Accessed: (A) 0=not accessed, 1=accessed

**Note:** If application segment, ( Bit 44 = 1 ), used to store program code and data.

Bit 43=0 is Data Segment

Bit 42: Expansion: 0=Expand Up, 1=Expand Down

Bit 41: Writeable: 0=Read Only, 1=Read/Write

Bit 43=1 is Code Segment

Bit 42: Conforming: 0=Non-conforming, 1=Conforming

Bit 41: Readable: 0=Execute Only, 1=Read/Execute

**Note:** If system segment, ( Bit 44 = 0 )

Bits 39-42 Type of segment

00	RESERVED
01	Available 286 TSS ( 16-bit )
02	LDT
03	Busy 286 TSS ( 16-bit )
04	286 Call Gate ( 16-bit ) ( Parm Count is words )
05	Task Gate
06	286 Interrupt Gate ( 16-bit )
07	286 Trap Gate ( 16-bit )
08	RESERVED
09	Available 386 TSS ( 32-bit )
10	RESERVED
11	Busy 386 TSS ( 32-bit )
12	386 Call Gate ( 32-bit ) ( Parm count is doublewords )
13	RESERVED
14	386 Interrupt Gate (32-bit )
15	386 Task Gate ( 32-bit )

-----

## Descriptor Table Summary

There are three descriptor tables at any instant.

1. [Global Descriptor Table](#)

located via GDTR

1 per system

accessible to all processes

describes objects common to all processes

2. [Local Descriptor Table](#)

LDTR is selector

GDT Descriptor Locates LDT

1 per process except for VDMs in which multiple LDTs are possible.

describes data unique to one process

3. [Interrupt Descriptor Table](#)

located via IDTR

1 per system except for VDMs in which multiple IDTs are possible.

describes interrupt routine entry points

-----

# The Selector Registers

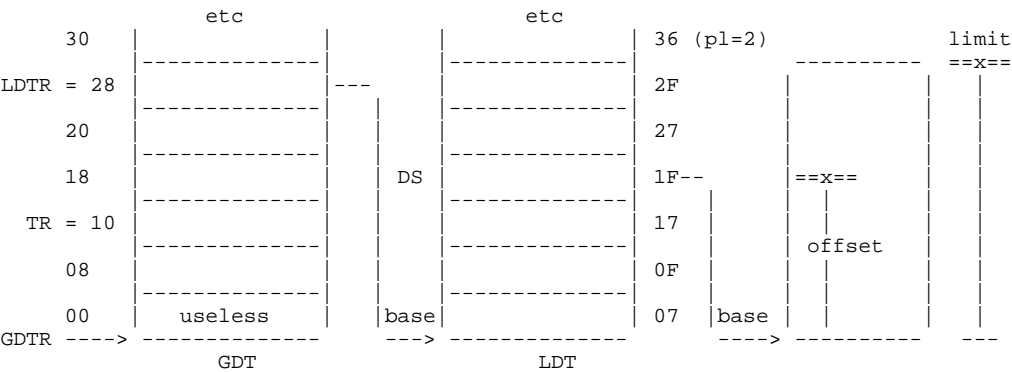
Each selector register appears to be 16 bits long. The six application selector registers and a brief description of the use for each follow:

- SS: Stack Selector, specifies the descriptor used for stack references.
  - CS: Code Selector, specifies the descriptor used for instruction references.
  - DS: Data Selector, specifies the descriptor used for most data references.
  - ES: Extra Selector, specifies another descriptor used for data references.
  - FS: This is a selector which can be used for data references if explicitly specified.
  - GS: This is a selector which can be used for data references if explicitly specified.
- The two system selector registers and a brief description of the use for each follow:
- LDTR: The LDT register selects the LDT descriptor from the GDT.
  - TR: The Task Register selects the descriptor used for the TSS.

## When Checking Is Done

When a program moves data into a selector register, that data becomes a selector and the processor fetches the content of the appropriate entry from the specified table into onboard registers which are not accessible to the programmer. The processor verifies the validity of the attempted access to the memory whenever a selector register is updated. This makes the protection overhead occur as part of the instruction which modifies a selector register, but eliminates it for further use of the selector.

**Note:** If the RPL of the SS register is not the same as CPL, or if an attempt is made to move the null selector into SS, a general protection exception occurs.



**Note:** The first descriptor in the GDT is reserved, by definition, and cannot be used. Any selector which would reference it is called the NULL SELECTOR; possible values are 0000, 0001, 0002, and 0003.

By definition, the null selector may be placed in DS, ES, FS, or GS, but any attempt to form an address with it is a general protection fault.

**Note:** The LDTR is a register that contains a selector. It can be accessed only by privilege level 0 instructions. It must contain a selector that references the GDT, and a descriptor whose "type" is LDT.

**Note:** It is not unusual for a GDT selector to describe the same storage as an LDT selector does. In OS2 2.x, application selectors in the GDT happen to describe one 448 Meg segment, not just a 64K segment like the LDT selectors describe. The linear address assigned to each LDT descriptor is extremely convenient for changing one form of an address to another, called thunking, which will be discussed later.

# Descriptor Examples

These examples come from DUMP1, which is used for several exercises.

```
DL 7 37
0007 Data Bas=ac6d7000 Lim=0000ffff DPL=3 P RO
000f Code Bas=00010000 Lim=00002e77 DPL=3 P RE A
0017 Data Bas=00020000 Lim=0000290f DPL=3 P RW A
001f Data Bas=00030000 Lim=000018af DPL=3 P RW A
0027 Data Bas=00040000 Lim=0000030a DPL=3 P RW A
002f Data Bas=00050000 Lim=00000fff DPL=3 P RW
0036 Data Bas=00060000 Lim=00000fff DPL=2 P RW A

DL BECF
bece Code Bas=17d90000 Lim=00000010 DPL=2 P RE A

DL BFD7 BFEF
bfd7 Data Bas=17fa0000 Lim=0000ffff DPL=3 P RW A
bfd7 Data Bas=17fb0000 Lim=0000ffff DPL=3 P RW A
bfee Code Bas=17fd0000 Lim=00000aa2 DPL=2 P RE A

DG 20 78
0020 Data Bas=ffe5b000 Lim=000003ff DPL=0 P RW UV
0028 LDT Bas=ac6d7000 Lim=0000ffff DPL=0 P
0030 Data Bas=ffe09de4 Lim=0000421b DPL=0 P RW ED A UV
003b Data Bas=ff4cbe2c Lim=00000073 DPL=3 P RW
0040 Data Bas=ffe5a400 Lim=000003bf DPL=0 P RW UV
004a Data Bas=00000000 Lim=1bfffffff DPL=2 P RW A G4k BIG UV
0053 Data Bas=00000000 Lim=1bfffffff DPL=3 P RW A G4k BIG UV
005a Code Bas=00000000 Lim=1bfffffff DPL=2 P RE C A G4k C32 UV
0063 Data Bas=00000000 Lim=1fffffff DPL=3 P RW G4k BIG UV
006b Data Bas=00000000 Lim=1bfffffff DPL=3 P RW A G4k BIG UV
0070 Data Bas=ffe22000 Lim=000074e4 DPL=0 P RO A
0078 Data Bas=ffe22000 Lim=000074e4 DPL=0 P RW

DG 148 L 4
0148 Code Bas=fff39000 Lim=00009262 DPL=0 P RE A
0150 Code Bas=fff43000 Lim=0000e137 DPL=0 P RE A
0158 Data Bas=00000000 Lim=ffffffff DPL=0 P RW A G4k BIG
0160 Code Bas=00000000 Lim=ffffffff DPL=0 P RE A G4k C32
```

The top section of the above output was created by entering the command `DL 7 37`

By inspecting the type, base, and limit fields in the above output, we can see the following about the descriptor referenced by 002F:

The storage is described as data having a base, or linear, address of 00050000. The linear address is not normally written with leading zeros. If there were any chance that the address might be mistaken for physical, a percent sign would be used, for example, %50000. The limit is FFF, which means that the segment is 4K, or 1000(hex) long. The privilege level is 3, the segment is present, and the flags indicate Read/Write storage. It has NOT been accessed, because the 'A' flag is not present, and OS/2 no longer uses this flag; once set by the hardware, it remains set.

Examples related to privilege level protection follow below:

CS:IP	CPL	DS:xxxx	RPL	lesser privilege CPL & RPL	DPL (from descriptor)	Access allowed?
000F:xxxx	3	17:xxxx	3	3	3	Yes
000F:xxxx	3	16:xxxx	2	3	3	Yes
000F:xxxx	3	14:xxxx	0	3	3	Yes
000F:xxxx	3	37:xxxx	3	3	2	No
000F:xxxx	3	36:xxxx	2	3	2	No
000F:xxxx	3	34:xxxx	0	3	2	No
000F:xxxx	3	43:xxxx	3	3	0	No
000F:xxxx	3	42:xxxx	2	3	0	No
000F:xxxx	3	40:xxxx	0	3	0	No
BECE:xxxx	2	17:xxxx	3	3	3	Yes
BECE:xxxx	2	16:xxxx	2	2	3	Yes

BECE:xxxx	2	14:xxxx	0	2	3	Yes
BECE:xxxx	2	37:xxxx	3	3	2	No
BECE:xxxx	2	36:xxxx	2	2	2	Yes
BECE:xxxx	2	34:xxxx	0	2	2	Yes
BECE:xxxx	2	43:xxxx	3	3	0	No
BECE:xxxx	2	42:xxxx	2	2	0	No
BECE:xxxx	2	40:xxxx	0	2	0	No
0150:xxxx	0	17:xxxx	3	3	3	Yes
0150:xxxx	0	16:xxxx	2	2	3	Yes
0150:xxxx	0	14:xxxx	0	0	3	Yes
0150:xxxx	0	37:xxxx	3	3	2	No
0150:xxxx	0	36:xxxx	2	2	2	Yes
0150:xxxx	0	34:xxxx	0	0	2	Yes
0150:xxxx	0	43:xxxx	3	3	0	No
0150:xxxx	0	42:xxxx	2	2	0	No
0150:xxxx	0	40:xxxx	0	0	0	Yes

In each case, as you read across you will see that CPL comes from the value of the CS register, RPL comes from the two low-order bits of the selector, and DPL comes from the descriptor. The column titled 'lesser privilege' is calculated remembering that higher numbers are lower privilege. The final column is obtained by following the access rules, a short way back.

## Exercise 1: Selectors and Descriptors

Objectives:

1. Learn how to load a dump for analysis
2. Introduction to the dump formatter
3. Learn how to display descriptors

Start the lab at a full-screen or windowed command prompt.

A full-screen session is faster, but a windowed session can be made 100 lines high by entering

MODE CO80,100 This can be very useful, because you can look back quite a ways by using the scroll bar.

Change to directory CLASSES\UTIL

Make diskette one by typing OS2IMAGE ..\IMAGES.162\LAB01.001 A:

Make diskette two by typing OS2IMAGE ..\IMAGES.162\LAB01.002 A:

Load the dump into a new file which will be named DUMP1.DMP by typing

DCOMP A: X:\DUMP01.DMP and pressing enter, then following the prompts.

When the dump is loaded, it should have a file size of 4194816.

Start the dump formatter by typing DF\_RET X:\DUMP01.DMP,

or by DF\_RET ..\DUMPS.162\DUMP01.DMP

You should see 6 or 7 informational lines at the top, followed by a pair of lines which start "Slot", and "0023#", followed by a set of registers.  
\*\*\* We are not yet concerned with any of these. \*\*\*

You should get a prompt, which is the character "#".

**Note:** You can always document what you are thinking by simply typing it in as an evaluation for the dump formatter to perform. You can access the evaluation function by typing '?' followed by whatever you want echoed to the screen and to the log. You can also type in ? and any expression to have it evaluated and output in hex, decimal, octal(!), binary, character, and boolean forms.

**Note:** ? by itself is a simple request for what commands are recognized.

Use the dump formatter to look at descriptors and answer these questions.

The dump formatter is NOT case sensitive.

Descriptors may be displayed using "DG" or "DL", followed by the selector. Try it both ways for several selectors, such as F, 160, DFFF, 158.

Use the miniature command reference in the back of the student guide, if necessary.

There are a great many things we will NOT do in this exercise. We are using only a tiny part of the dump formatter's capabilities for this class. For example, we will ignore the IDT in this class; one can enter "DI" followed by the interrupt number to see the descriptor from the IDT.

Questions to answer:

1. Which table contains the descriptor data for selector 000F?
2. Which command is preferred to display only the descriptor for 000F?
3. What alternative command will also display only the descriptor for 000F?
4. What type of memory is described by selector 000F?

Hint: It is one of the first things displayed in the output for each descriptor.

5. What is the largest valid offset within segment 000F?
6. What is the size of segment 000F?

Hint: Not quite the same as the previous answer.

7. What is the linear ( virtual) address of segment 000F?
8. What privilege level is segment 000F?
9. What is the Requested Privilege Level of selector 000F?

Hint: RPL is not in the descriptor.

10. What is the type and limit of segment 0017?
11. What is the linear (virtual) address of segment 0017?
12. Which table contains descriptor 0017?
13. Will the application program be able to access the segment selected by 000F?

Explain.\_\_\_\_\_

14. Will the program be able to store into segment 000F?

Explain.\_\_\_\_\_

15. Will the application program be able to access storage using selector 0037?

Explain.\_\_\_\_\_

16. Will the program be able to write into storage using selector 38?

Explain.\_\_\_\_\_

17. Will the program be able to write into storage using selector 0007?

Explain.\_\_\_\_\_

18. Enter the following command: DG 70 L 2

Compare and contrast the base, limit, privilege level and flags for each.

19. Enter the following command: DG 5A;DG 5B

Compare and contrast the base, limit, privilege level and flags for each.

20. Enter the following command: DG 28;DL 7

Compare and contrast the base, limit, privilege level and flags for each.

The dump formatter will exit in response to the command 'Q'.

# Address Mapping

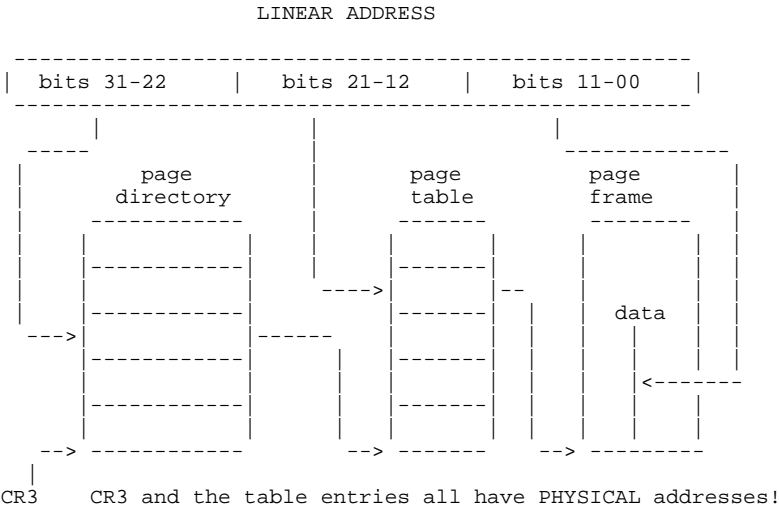
This section describes the method used to transform addresses from linear addresses to physical addresses.

## Paging Overview

OS/2 V2 uses paging in addition to the above logical addressing. Paging is a mechanism which converts linear addresses to physical addresses and allows a consistent size ( 4k ) to be moved back and forth to auxiliary storage ( SWAPPER.DAT ) when the demand for virtual memory exceeds the physical memory installed on the machine. Another hardware register, Control Register 3 or CR3, is used to locate a page directory which contains table entries that locate page tables. The page tables are used to locate the physical memory where the data really resides. Physical memory is sometimes referred to by page number. A page number is simply the twenty high-order bits of an address. The twelve low-order bits of a page address are all zero. One can convert a page number to an address by simply appending three hex zeros to it.

THE RESULT OF COMBINING A SEGMENT NUMBER AND AN OFFSET, OR THE ADDITION OF AN OFFSET TO THE BASE ADDRESS FROM A DESCRIPTOR, IS A LINEAR ADDRESS. Under OS/2 1.x, these would be physical addresses. Under OS/2 2.0 and following, these are linear, or virtual addresses. The picture below shows how linear addresses are converted to physical addresses. Only the top line in the picture below is a linear address - the rest are physical.

The ten high order bits of the linear address are used to index into the Page Directory which has the twenty high order bits of the page table's physical address (page number). The next ten bits of the linear address are used to index into the page table. The twenty high order bits of the page frame's physical address (page number) are retrieved. The twelve low order bits of the linear address are also the twelve low order bits of the physical address. Therefore, the physical address is the twenty bits from the page table entry, followed by the 12 low-order bits from the linear address.



## Page Table Entries

The page directory entries are identical to the page table entries.



Each entry is 4 bytes, making 1K entries in each page table.

Bits 31-12	Physical Address of page, or Page Frame Address
Bits 11-09	Ignored by hardware, used by OS2. See Note.
Bits 08-07	Reserved, must be zero
Bit 6	Dirty (D) 0=not changed (clean), 1=changed (dirty)
Bit 5	Accessed (A) 0=not accessed, 1=accessed
Bit 4	Page Cache Disable 0=allow cache use, 1=bypass cache
Bit 3	Page Write-Through 0=cache write-into, 1=write through to RAM
Bit 2	Supervisor (S/U) 0=Supervisor (PL=0,1,2), 1=user (PL=3)
Bit 1	Write enable (RO/RW) 0=Read Only, 1=Read/Write
Bit 0	Present (P) 0=not present, 1=present

**Note:** The left 5 hex digits of the entry are the left 5 hex digits of the physical page; while the right 3 hex digits are mostly flags.

**Note:** If Bit 0 is zero, ( page invalid ) the remaining bits are NOT inspected by the hardware. OS/2 uses them to identify the virtual page associated with this address.

**Note:** Bits 09 and 10 are used to track the state of the page frame. Three of the possible four combinations are used:

0 - Pageable

1 - [UVirt](#)

2 - Resident

## Page Table Contents

To look at the contents of the page directory and page table(s), use the command `DP`, followed by the address of interest.

```
DP F:0
  linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%00010000* 001e0  frame=0009e  0    0  c  A          U  r  P  pageable
%00010000  0009e  frame=0009e  0    0  c  A          U  r  P  pageable
%00011000   vp id=012ae  0    0  c  u          U  r  n  pageable
%00012000  00292  frame=00292  0    0  c  A          U  r  P  pageable

DP 17:0
  linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%00020000* 001e0  frame=00181  0    0  D  A          U  W  P  pageable
%00020000  00181  frame=00181  0    0  D  A          U  W  P  pageable
%00021000  003d4  frame=003d4  0    0  D  A          U  W  P  pageable
%00022000  0005a  frame=0005a  0    0  D  A          U  W  P  pageable

DP 1F:0
  linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%00030000* 001e0  frame=003ae  0    0  D  A          U  W  P  pageable
%00030000  003ae  frame=003ae  0    0  D  A          U  W  P  pageable
%00031000  001b5  frame=001b5  0    0  D  A          U  W  P  pageable

DP 27:0
  linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%00040000* 001e0  frame=00052  0    0  c  A          U  W  P  pageable
%00040000  00052  frame=00052  0    0  c  A          U  W  P  pageable

DP 2F:0
  linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%00050000* 001e0  frame=00075  0    0  D  A          U  W  P  pageable
%00050000  00075  frame=00075  0    0  D  A          U  W  P  pageable
```

```
DP 37:0
linaddr    frame    pteframe    state    res    Dc    Au    CD    WT    Us    rW    Pn    state
%00060000* 001e0    frame=002ae    0        0    D    A        U    W    P    pageable
%00060000    002ae    frame=002ae    0        0    D    A        U    W    P    pageable
```

In each case, the first line of output is the data from the page directory.

The field labelled 'frame' is the physical page frame which holds the data at the referenced address.

The 'vp id' is the virtual page identifier for the entry %11000.

'Dc' is Dirty or Clean. 'Au' is Accessed or unaccessed.

'Us' is User (Ring 3) or supervisor (Rings 0 & 2).

'rW' indicates read-only or Writeable. 'Pn' indicates Present or not-present.

## Data Format in Storage

Data format is least significant byte at lowest address!

This arrangement is not intuitive for many people, because when you read bytes, the data placement seems reversed. The tools will let you display storage as bytes, words, and doublewords; the data will be re-arranged to suit the format requested. This can be good or bad.

For example:

```
DB 1F:1608 L 20
001f:00001608 42 4f 4f 4b 53 48 45 4c-46 3d 43 3a 5c 4f 53 32 BOOKSHELF=C:\OS2
001f:00001618 5c 42 4f 4f 4b 3b 00 43-4f 4d 53 50 45 43 3d 43 \BOOK;.COMSPEC=C

DA 1F:1608
001f:00001608 BOOKSHELF=C:\OS2\BOOK;

DB 17:0 L40
0017:00000000 02 00 03 00 05 00 07 00-0b 00 0d 00 11 00 13 00 .....
0017:00000010 17 00 1d 00 1f 00 25 00-29 00 2b 00 2f 00 35 00 .....%.).+./..5.
0017:00000020 3b 00 3d 00 43 00 47 00-49 00 4f 00 53 00 59 00 ;.=.C.G.I.O.S.Y.
0017:00000030 61 00 65 00 67 00 6b 00-6d 00 71 00 7f 00 83 00 a.e.g.k.m.q.....

DW 17:0 L20
0017:00000000 0002 0003 0005 0007 000b 000d 0011 0013
0017:00000010 0017 001d 001f 0025 0029 002b 002f 0035
0017:00000020 003b 003d 0043 0047 0049 004f 0053 0059
0017:00000030 0061 0065 0067 006b 006d 0071 007f 0083

DW 17:1 L 20
0017:00000001 0300 0500 0700 0b00 0d00 1100 1300 1700
0017:00000011 1d00 1f00 2500 2900 2b00 2f00 3500 3b00
0017:00000021 3d00 4300 4700 4900 4f00 5300 5900 6100
0017:00000031 6500 6700 6b00 6d00 7100 7f00 8300 8900

DD 17:0 L 10
0017:00000000 00030002 00070005 000d000b 00130011
0017:00000010 001d0017 0025001f 002b0029 0035002f
0017:00000020 003d003b 00470043 004f0049 00590053
0017:00000030 00650061 006b0067 0071006d 0083007f

DD 17:1 L 10
0017:00000001 05000300 0b000700 11000d00 17001300
0017:00000011 1f001d00 29002500 2f002b00 3b003500
0017:00000021 43003d00 49004700 53004f00 61005900
0017:00000031 67006500 6d006b00 7f007100 89008300

DD 17:2 L10
0017:00000002 00050003 000b0007 0011000d 00170013
0017:00000012 001f001d 00290025 002f002b 003b0035
0017:00000022 0043003d 00490047 0053004f 00610059
0017:00000032 00670065 006d006b 007f0071 00890083
```

You need to know what you are looking at!

## Exercise 2: Paging, Addresses, Data

Objectives:

1. Reinforce the knowledge from exercise 1
2. Learn how to display page table data
3. Learn how to convert a logical address to a linear address
4. Learn how to convert a linear address to a physical address
5. Learn how to display storage as ASCII, bytes, words, and doublewords.

Startup directions:

1. Start the dump formatter by typing `DF_RET ..\DUMPS.162\DUMP01.DMP`
2. You should see the standard startup messages.
3. The initial register display is what the application registers were at the time the application (ring 3) program trapped.
4. You can see these at any time by entering the ".R" command.
5. Use the dump formatter to look at the dump and answer these questions.  
The dump formatter is NOT case sensitive.

**Note:** Paging data may be displayed using the "DP" command, followed by the address.

**Note:** The dump process DESTROYS the first entry of the page directory. You will get quite confused if you try to follow the hardware method to look at paging information for addresses 0 - 3FFFFFF.

If you must, use the '.N' command to find "savepage", which will tell you the physical address of the page table for that address range.

This may well be the last time you use a physical address in an OS/2 debugging session. With the notable exceptions of physical memory management and physical device drivers, OS/2 is almost completely unaware of physical addresses. The 32-bit virtual address, also called a linear address, and a 'flat' address, is what is used in general throughout OS/2.

Assuming these registers, answer the following questions:

```
eax=0000c8cf ebx=00002910 ecx=000000df edx=00000000 esi=00000030 edi=00000060
eip=000000be esp=000014be ebp=000014e6 iopl=2   rf -- -- nv up ei pl zr na pe nc
cs=000f  ss=001f  ds=001f  es=0017  fs=150b  gs=0000   cr2=00000000  cr3=001a7000
```

1. What are the base and limit fields for selector 000F? (the base is the linear address...)
2. How many 4k pages are in this segment? Hint: Look closely at the limit field.
3. How many physical pages are allocated for the virtual memory segment starting at F:0?

Hint: `DP 0F:0` or `DP %10000`

4. Why are the above two answers different?
5. What is the physical address of the data at F:0?

Observation: You now have three ways to address the data.

- a. Real or V86 (&selector:offset)
- b. Logical (#selector:offset)
- c. Linear (%address)

d. Physical (%%address)

We will now display the same storage many ways, to confirm we know how.

6. What is the command to display the storage at SS:BP in words using a logical address?
7. What is the command to display the storage at SS:BP in words using a linear address?
8. What is the command to display the storage at SS:BP in words using a physical address?

For each of the following, study the results until you understand.

9. Display the data at 7:0 as bytes, and words.
10. Display the data at 7:1 as bytes and words.
11. Display the data at 7:0 and 7:1 as words.
12. Display the data at 7:0 as words and doublewords.
13. Display the data at 1F:15C6 as bytes and ASCII. Also look at 1F:15DA as bytes and ASCII.

-----

## Instruction Set

This section discusses the '86 registers & some common instructions from the instruction set.

-----

## Register Review

Registers discussed so far:

CR3	32-bit physical address of the Page Directory
IDTR	32-bit linear address of IDT, 16-bit size of IDT
GDTR	32-bit linear address of GDT, 16-bit size of GDT
LDTR	16-bit selector for an entry ( type 2 ) in the GDT
SS	16-bit selector, used for stack operations
CS	16-bit selector, used to locate instructions
DS	16-bit selector, used to locate data, generally the default
ES	16-bit selector, used to locate data, string destination
FS	16-bit selector, used to locate data explicitly
GS	16-bit selector, used to locate data explicitly

-----

## Execution

386 execution consists of the classic pattern of fetching an instruction from memory and executing it, then repeating the process. The instructions are always found in a code segment accessed via the descriptor designated by the selector in the CS register. The current privilege level of the program is contained by the two low order bits in the CS register. The offset of the next instruction is contained in the

instruction pointer, (IP or EIP) which is incremented as each instruction is fetched. The 386 and following generations recognize a great number of instructions, but compilers generate a very small subset of the whole instruction set. Much of that subset will be discussed here. If you cannot ascertain what an instruction does when you encounter it, look it up in the appropriate reference manual. Instructions are generally executed sequentially, and the processor attempts to fetch instructions well in advance, to increase execution speed. The flow of control departs from sequential when a jump, call, return, interrupt or interrupt return is encountered. Jumps are conditional or unconditional. Conditional jumps are used to implement decisions and contain a relative offset which is combined with IP by signed addition to cause a different instruction in the same segment to be executed next. Calls, returns and unconditional jumps come in two varieties: NEAR and FAR. The NEAR variety update only IP and leave CS untouched. The FAR variety update both CS and IP and are potentially quite complex. CALL, RETurn and interrupts require a stack. Most instructions reference the registers.

## General Registers

	EAX		ALL 32 BITS
( part of EAX )		AX	LOW 16 BITS
( part of AX )		AH	HIGH 8 BITS
( part of AX )		AL	LOW 8 BITS

Registers EBX, ECX, and EDX also subset in the same way.  
There are two byte-sized pieces, which can be collectively referenced as a word-sized item.

	EIP		ALL 32 BITS
( part of EIP )		IP	LOW 16 BITS

IP and EIP are always offsets into CS.  
They always contain the address of the next instruction to execute.

	ESP		ALL 32 BITS
( part of ESP )		SP	LOW 16 BITS

SP and ESP are always offsets into SS.  
They contain the address of the last item pushed into the stack.

REGISTERS EBP, ESI, and EDI also subset in this way.  
They have no 8 bit parts.

## Machine Instructions

There are several fields which may be present in an instruction. Additionally, there are a few easy-to-learn generalities which will make understanding what an instruction does much easier. Data definitions will not be covered here. There are many fields possibly present in an instruction.

1. The label.

The label is optional, but must be first. It is followed by a colon. It is used so the programmer can refer to the instruction symbolically. A label does not require an instruction.

Labels which are 'Public' become symbols at link time.

2. The mnemonic operation code, or opcode, is next.

It defines what operation will be attempted, and therefore what operands need to be specified. Instructions have zero to three specified operands; many instructions also imply operands.

3. The operands are next, separated by commas.

The first operand is always the result, or target, of the operation.

An operand may be a value, a register, or storage. When the operand is a value, it is called 'immediate', because the operand is immediately available if the instruction has been fetched. When a register is named, it is the operand. If an expression is contained in brackets, it is evaluated and the result is used as a offset into some segment.

A storage operand is in some segment by default. Data references default to the data segment, or DS, unless (E)BP or (E)SP are present in the address expression. In this case, the default segment is the stack segment (SS). (E)IP is ALWAYS in the CODE segment (instructions). (E)SP is ALWAYS in the STACK segment (data). (E)BP is USUALLY in the STACK segment (data).

The default segment can usually be overridden by specifying the selector as part of the address, for example, DS:[BP+8].

You will come across helper words within operands, such as "byte", "word", and "dword" which are there to remind you of the size of the data item referenced. You will also come across the helper word "ptr", which is to remind you that the addressed data is in storage, and that the offset, in brackets, is a pointer to the data.

4. The last item you may find is an optional comment.

A comment is preceeded by a semicolon. Anything following is a comment. Comments are sparse in the output of the 'Unassemble' command.

The debug kernel will use a comment to identify a breakpoint.

Both the debug kernel and the dump formatter will supply a symbol anytime a number matches the symbol in an active file.

-----

## Typical Instructions

MOV CL,DH

The opcode is 'MOV', the first operand is the CL register, and the second operand is the DH register. This instruction will copy (MOVE) all 8 bits from the DH register to the CL register.

MOV DX,8

The opcode is 'MOV', the first operand is the DX register, the second operand is the immediate value of 8. This instruction puts the value 8 into the DX register.

MOV EBP,ESP

Again, the opcode is 'MOV', and the instruction will copy all 32 bits of ESP into EBP.

MOV AX,BX

You should be able to tell by now that this instruction will copy 16 bits from BX to AX. Note that instructions which reference only registers are extremely unlikely to cause an exception.

MOV AX,word ptr [BX]

This instruction is different from the one above because there are brackets around the second operand. This means that the operand, BX in this case, is in storage, and the BX register holds the offset into the DS segment. If BX is outside the limit of the DS segment, a general protection fault will occur.

MOV word ptr [BX],AX

This instruction is similar to the preceding one, but moves data into storage, rather than from storage. The same exceptions might occur, and if the DS segment is read-only, this instruction would also fail.

MOV word ptr ES:[BX],DI

This is an example of overriding the default segment, DS, by explicitly specifying that the offset in the BX register applies to the ES segment.

ADD word ptr DS:[BP],AX

This would add the 16 bits from AX into storage at DS:BP, developing the sum directly in storage. The override is needed because the use of BP means that the default segment is SS.

DEC word ptr [BP-2]

Some instructions have only one operand. In this case it is in storage at an offset calculated by subtracting 2 from the BP value, in the segment defined by the SS register, because BP is used.

Also SUB, CMP, AND, OR, XOR, XCHG, INC, SHL, etc.

It is extremely common for 16-bit code to use FAR addresses. When they are in storage, it would require several instructions to get a FAR address into the registers, if it were not for several instructions whose purpose is specifically to fetch a FAR address from storage into a selector and another register. These instructions may be recognized by the opcode, which is the letter 'L' followed by a selector register name other than CS. The apparent first operand is the general, base, or index register which will hold the offset part of the far address. Both registers will be loaded, with the first operand coming from the address specified, and the selector coming from the following word.

LES BX,dword ptr [BP+6]

This instruction loads BOTH BX and ES. BX comes from BP+6 and ES comes from BP+8, both in the stack segment.

LDS SI,dword ptr [BP-12]

This instruction loads BOTH SI and DS. SI is loaded from BP-12 and DS is loaded from BP-10.

LEA EDI,[EBP+ECX\*4-12]

Load Effective Address DOES NOT actually reference storage. Instead, once the offset has been calculated, it is put into the target register, EDI in this case. Address expressions like this are possible, but not often seen while actually debugging. The scale factor can be 1, 2, 4, or 8; not any arbitrary value

---

## The System Flags (EFLAGS Register)

The flags, which are contained in the EFLAGS register, not only control system operation, but also hold the result of instructions such as CMP (compare). At times, you will find the flags have been copied to a register, or to memory. The following figure gives the format of the flags in such cases:

Bit	Hex	Flag name	Comments
18	00040000	AC	Alignment Check, if the alignment mask is 1 (CR0).
17	00020000	VM	V86 mode. Turned on for Virtual DOS Machines.
16	00010000	RF	Resume Flag. Suppress debug exceptions for 1 instruction.
14	00004000	NT	Nested Task. Involved with hardware task switching.
13/12	3000	IOPL	The least privileged code which has unrestricted I/O access.
11	0800	OF	Overflow. An arithmetic result does not fit.
10	0400	DF	Direction of string instructions. 0=up, 1=down.
09	0200	IF	Interrupt flag. 1=enabled, 0=disabled.
08	0100	TF	Trap flag. Generate a debug exception after each instruction.
07	0080	SF	Sign. 1=minus, 0=plus.
06	0040	ZF	Zero or Equal. 1=zero result, 0=non-zero result.
04	0010	AF	Auxiliary flag. Used in BCD arithmetic.
02	0004	PF	Parity flag. 0=even, 1=odd.
00	0001	CF	Carry flag. 0=no carry, 1=carry.

---

# Unassembled Instructions

```
U CS:IP-22 IP-18
000f:0000009c f1          db      f1
000f:0000009d 8946fc        mov     word ptr [bp-04],ax
000f:000000a0 f7e1          mul     cx
000f:000000a2 8946f4        mov     word ptr [bp-0c],ax
000f:000000a5 3946f6        cmp     word ptr [bp-0a],ax

U CS:IP-23 IP-18
000f:0000009b f7f1          div     cx
000f:0000009d 8946fc        mov     word ptr [bp-04],ax
000f:000000a0 f7e1          mul     cx
000f:000000a2 8946f4        mov     word ptr [bp-0c],ax
000f:000000a5 3946f6        cmp     word ptr [bp-0a],ax

U CS:IP-24 IP-18
000f:0000009a ee          out     dx,al
000f:0000009b f7f1          div     cx
000f:0000009d 8946fc        mov     word ptr [bp-04],ax
000f:000000a0 f7e1          mul     cx
000f:000000a2 8946f4        mov     word ptr [bp-0c],ax
000f:000000a5 3946f6        cmp     word ptr [bp-0a],ax

U CS:IP-25 IP-18
000f:00000099 56          push    si
000f:0000009a ee          out     dx,al
000f:0000009b f7f1          div     cx
000f:0000009d 8946fc        mov     word ptr [bp-04],ax
000f:000000a0 f7e1          mul     cx
000f:000000a2 8946f4        mov     word ptr [bp-0c],ax
000f:000000a5 3946f6        cmp     word ptr [bp-0a],ax

U CS:IP-26 IP-18
000f:00000098 8b56ee        mov     dx,word ptr [bp-12]
000f:0000009b f7f1          div     cx
000f:0000009d 8946fc        mov     word ptr [bp-04],ax
000f:000000a0 f7e1          mul     cx
000f:000000a2 8946f4        mov     word ptr [bp-0c],ax
000f:000000a5 3946f6        cmp     word ptr [bp-0a],ax

U CS:IP-27 IP-18
000f:00000097 ec          in      al,dx
000f:00000098 8b56ee        mov     dx,word ptr [bp-12]
000f:0000009b f7f1          div     cx
000f:0000009d 8946fc        mov     word ptr [bp-04],ax
000f:000000a0 f7e1          mul     cx
000f:000000a2 8946f4        mov     word ptr [bp-0c],ax
000f:000000a5 3946f6        cmp     word ptr [bp-0a],ax

U CS:IP-10 IP
000f:000000ae 3976f0        cmp     word ptr [bp-10],si
000f:000000b1 77df          ja      0092
000f:000000b3 3976f0        cmp     word ptr [bp-10],si
000f:000000b6 7510          jnz     00c8
000f:000000b8 c45ede        les     bx,dword ptr [bp-22]
000f:000000bb 8b46f6        mov     ax,word ptr [bp-0a]
000f:000000be 268907        mov     word ptr es:[bx],ax
```

---

## Observations About Unassembling From an Unknown Starting Place

Instructions are of variable length, from one to fifteen bytes long.



This means the address you provided may not actually be the start of an instruction. This also means, therefore, that the first few instructions you see may not actually be what the machine saw.

If you look at the output of several unassemblies starting at sequential addresses, you will see that after typically 3 to 5 tries, the unassembly will agree with previous ones, for some point after the unassembly started.

This is typically within four or five lines, but not always. Be cautious, and see if the sequence looks reasonable. If it does, you have most likely found an instruction boundary. Experience will help this process.

Some common sense will help as well. Obviously, an application in ring 3 cannot perform I/O directly. Likewise, the 'db' means that the unassembler did not have a way to interpret this as an instruction.

The last command entered looks at a few of the instructions which actually preceeded a failure.

Can you discover which instruction put the data into the ES and BX registers?

-----

## Exercise 3: Unassembling and Reading Instructions

Objectives:

1. Reinforce the preceeding lab exercises
2. Learn how to unassemble instructions
3. Learn how to read instructions
4. Learn about variable length instructions

We will now look at instructions.

1. In what type of segment are instructions found?
2. Are instructions EVER executed in any other segment type?
3. Unassemble the instructions which would have been next to execute ( if the application hadn't trapped ) by entering "U". The default address is CS:IP initially. You can unassemble further with repeated use of "U". To unassemble at a particular place, specify the address; for example CS:IP.
4. What was the next instruction which would have executed?
5. Unassemble using an address range to see some previous instructions. Type "U CS:IP-20 IP-10". This will unassemble from ip-20 to ip-10. Now type "U CS:IP-21 IP-10" and "U CS:IP-22 IP-10. Observe what is happening by closely observing the address at which each instruction begins.
6. Now type "U CS:IP-18 IP" to see the TWO instructions immediately before the failing instruction

( at CS:IP! ). What are they?

7. Which one loaded the address used in the next (failing) instruction?
8. Did the address come from this routine's private data, or was it a parameter passed by the caller?

This is presented in detail later.

9. Circumstantially at least, what seems to be wrong?

Also presented later.

This page left mostly blank so that the next pair of pages will face each other.

-----

## Exceptions

Events sometimes occur which disrupt the normal sequence of instruction. These are called exceptions and interrupts. Intel defines exceptions in relation to an unsuccessful attempt to execute an instruction. Interrupts are defined as a hardware response to a event unrelated to program execution.

# HEX	TYPE	B/C	ERR CODE	SOURCE CAUSE
0	Fault	C	No	Divide Overflow ( perhaps by zero )
1	DR6	B	No	Debug Exception
2	Int	B	No	NMI ( Non-Maskable Interrupt ), normally hardware fault
3	Trap	B	No	Breakpoint ( INT 3 instruction )
4	Trap	B	No	Overflow ( INTO instruction )
5	Fault	B	No	Bounds Check ( BOUND instruction )
6	Fault	B	No	Invalid Opcode
7	Fault	B	No	Coprocessor not available, see note
8	Abort	Abort	Always Zero	Double Fault, any instruction
9	Fault	C	Yes	Coprocessor Segment Overrun (286,386 only) (Fault D in 486+)
A	Fault	C	Yes	Invalid TSS
B	Fault	C	Yes	Segment Not Present ( swapped out )
C	Fault	C	Yes	Stack Exception
D	Fault	C	Yes	General Protection
E	Fault	PF	Yes	Page Fault ( paged out )
F				( reserved )
10	Fault	B	No	Coprocessor Error
11	Fault	?	Always Zero	Alignment Check
12	Abort	??	Machine Check	
13-1F				( reserved )
20-FF	Trap	N/A	No	Available for Hardware Interrupts Via 'INTR' Pin
00-FF	Trap	N/A	No	The INT instruction is actually a trap.

**Note:** Co-processor not available may be due to not having one, or because the content of the co-processor belongs to another thread. The co-processor data needs to be saved and restored only when more than one thread is using it. Bit 3 in CR0 indicates that a thread switch has occurred and will cause a trap 7 when a co-processor instruction is decoded.

Explanation of B/C column

B - Benign, means it is ok with any other exception

C - Contributory, means it will contribute to a double fault

PF - Page Fault, means a referenced address is not present

## Definition of Fault, Trap, Etc.

## 1. Faults

CS & EIP point to the instruction which generated the fault.

## 2. Traps

CS & EIP point to the instruction to be executed after the instruction which caused the trap.

INT3, INTO, BOUND, and INT nn are examples of traps.

## 3. Aborts

In general, these exceptions do not permit locating the failing instruction, nor restart of the thread which caused the abort. Aborts are used to report inconsistent or illegal values in system tables, and hardware errors.

## 4. Interrupts

Unlike the preceding exceptions, interrupts are not related to the program being executed, but to an external condition.

-----

# Hardware Error Codes

### Selector Related Error Code

Bits 31-15: Reserved.  
Bits 15-03: The index part of the selector involved.  
Bit 02: The table indicator bit,  
if neither bit 01 nor bit 00 are 1.  
Bit 01: IDT selector bit,  
if on, the selector is in the IDT.  
Bit 00: External bit,  
if on, not caused by the program

### Page Fault Error Code

Bits 31-04: Reserved.  
Bit 03: RSV. A 1 bit was detected in a reserved  
bit of a page directory or page table entry.  
Bit 02: U/S.  
0: The program was in supervisor mode.  
1: The program was in user mode.  
Bit 01: W/R.  
0: The access was a read.  
1: The access was a write.  
Bit 00: Level.  
0: The fault is because of a not-present page.  
1: The fault is because of page-level protection.

-----

# Simultaneous Exceptions

It is possible for more than one exception to occur while attempting to execute an instruction. In order to determine what will happen if two simultaneous exceptions occur on the same instruction, use the following table:

First	Second	Resulting
Exception	Exception	Action
Benign	Benign	OK
Benign	Contributory	OK
Benign	Page Fault	OK
Contributory	Benign	OK
Contributory	Contributory	Double Fault
Contributory	Page Fault	OK
Page Fault	Benign	OK
Page Fault	Contributory	Double Fault
Page Fault	Page Fault	Double Fault

**Note:** OK means the faults are processed consecutively.

**Note:** Double Fault means the faults are reported together.

**Note:** If any other exception occurs trying to enter the DoubleFault handler, the processor shuts down until RESET; although, if the NMI handler has not been entered, NMI will be recognized and accepted.

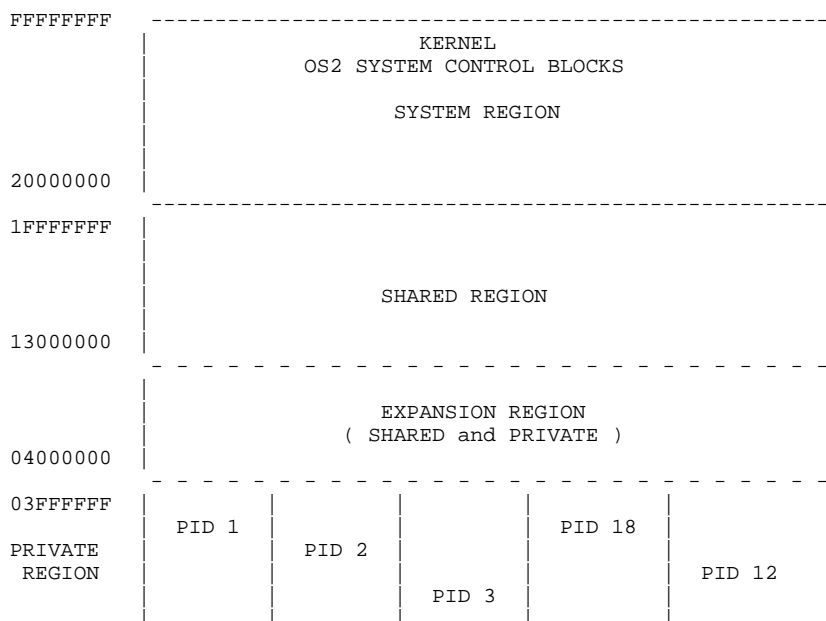
**Note:** A trap C in Ring 0 is usually a double fault.

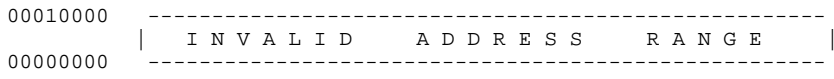
When the processor detects a Stack Exception it needs to push an error code and a return address onto the stack of the exception handler. If this happens in Ring 0, there will be no privilege level transition, which includes switching to a new, protected stack. If the exception is due to stack growth, there is no place to push the error code or return address.

RESULT: TRAP 8

# The Address Space Picture

This is a picture of what the address space looks like for several processes.





**Note:** Within the private region you must know the Process ID, as well as the linear address to define a piece of virtual storage. All regions except the private region are shared among all processes. Above the private region in the shared regions, there is only one version of a given address, so you DO NOT need the Process ID.

**Note:** The boundary at 03FFFFFF is an initial value. If some application allocates over 03FFFFFF of private space, this boundary will move upward. It moves in steps of 00400000, because another page table is allocated.

**Note:** DLL's are initially loaded beginning at the 1BFFFFFF boundary, and to successively lower addresses. This 'water mark' moves downward in steps of 00400000, too.

**Note:** Addresses not assigned to a memory object are invalid. Any attempt to use them will generate an exception.

**Note:** The address space picture discussed here is a simplified overview. A more detailed description may be found in the Advanced Guide to Hang Analysis chapter, under [Memory Management and Ownership Topics](#).

## OS/2 Implementation Details

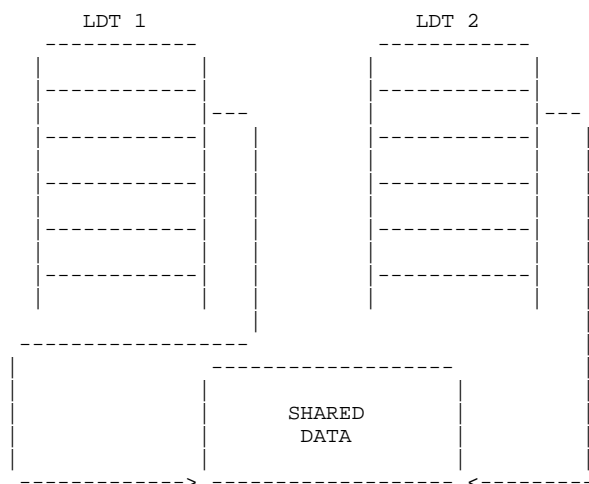
This section discusses some of the implementation details of OS/2 which particularly involve debugging.

## Shared Memory

This highlights how memory is shared among a few processes.

The same selector is allocated in each process that shares the storage. Each process therefore uses the same offset in the LDT, and the LDT entries are the same, so the same linear address is also used.

**Note:** The page table entries used for the shared storage are the same for both processes, too.



DLL's are a good example of shared storage.

DLLs are loaded into the shared address range. The boundary is dynamic, and moves downward as DLL's are loaded.

The boundary of private addresses move upward as private storage is allocated. There is a guarantee of 64 Meg for private, and 64 Meg for

shared.

---

## Address Tiling

Address tiling refers to the practice of creating a mathematical or algorithmic relationship between an LDT selector and the base, or virtual address in the descriptor.

By using address tiling, OS/2 avoids the need to move memory blocks because of reallocation, and also makes it very fast to convert an LDT Selector:Offset to a flat, or Linear Address. The implementation is simply to allocate 64K of virtual address space to each selector, starting with selector '000F', at virtual address 64K, or '%10000'.

**Note:** Selector '0007' is used to map the LDT as read-only data.

---

## Why Thunk?

It is still common to have applications which have some 32-bit parts, and some 16-bit parts. The 32-bit parts try to avoid using 16-bit selector:offset addressing, because of the overhead of loading the selector registers, as well as to avoid the challenge of correctly dealing with storage references in both modes.

A typical example is a 32-bit application calling a 16-bit DLL.

Since storage is (must be) the same for all parts of a process, there has to be a way to convert one form of an address to the other.

Only 16-bit application selectors from the LDT are eligible for this quick form of the conversion, and only linear addresses less than %20000000 can be converted to 16:16 format.

Additionally, addresses in the packed region may NOT be converted by this quick method, but by a search of the LDT descriptor base (linear) addresses, followed by a calculation.

The top of normal application space, at %1BFFFFFF, is mapped to selector DFFF. The top of protected shared addresses at %1FFFFFFF maps to selector FFFF, if used.

---

## Address Transformations (Thunks)

This section tells you how to change from 16:16 to 0:32-bit mode, or vice versa.

There are two parts to thunking, the address transformation, and properly aligning the stack, if necessary. The stack alignment is usually done by a subroutine which detects the need to do this, and builds an 'extra' frame in the new mode, properly aligned by making a copy of the incoming parms, transforming the addresses as part of this process.

This works only because the specific implementation within OS/2 which was designed to use address tiling for LDT selectors.

---

## 16:16 to 0:32 Thunk

The selectors which are eligible for this thunk are LDT selectors which are PL=3.

In this case, all three low-order bits are 1. Because of this, one can shift the selector three bits to the right, or divide by 8, without loss of information. The resulting number is the high-order word of the 32-bit address because of address tiling.

For example, address 000F:00BA can be thunked from 16:16 to 0:32 as follows:

```
0    0    0    F    :    0    0    B    A    <--- Hex Sel:Offset
0000 0000 0000 1111    0000 0000 1011 1010 <--- Binary
shift the selector 3 bits to the right, which gives
```

0000 0000 0000 0001                    0000 0000 1011 1010 <--- Binary  
0    0    0    1                    0    0    B    A    <--- Linear Address  
Note that the lower 16 bits, or offset, are unchanged.

A stack may require alignment, because a 32-bit stack is built on double-word boundaries, with two low order zero bits in the address of each element, whereas a 16-bit stack is aligned only on a word boundary.

## 0:32 to 16:16 Thunk

Because the range of LDT selectors is only 512 Meg, addresses less than this can be transformed to use an LDT selector, with restrictions. The transformation is to append three low-order 1 bits to the value, and to discard three high order zero bits. An alternative way of stating this is to multiply by 8, then add 7. The three low order one bits are LDT (table indicator=1), and PL=3. The restrictions are that the storage must be PL=3 application storage, must not span a 64K boundary in the linear address space, and the value must be less than hex 2000 0000.

0    0    0    2                    1    4    B    0    <--- Linear Address  
0000 0000 0000 0010                    0001 0100 1011 0000 <--- Binary  
shift the left half 3 bits to the left, which gives  
0000 0000 0001 0000                    0001 0100 1011 0000 <--- Binary  
add 7 to the left half ( 0111 binary )  
0000 0000 0001 0111                    0001 0100 1011 0000 <--- Binary  
0    0    1    7    :                    1    4    B    0    <--- Hex Sel:Offset  
Note that the lower 16 bits, or offset, are unchanged.

## Simultaneous 16-bit and 32-bit Descriptions of Virtual Storage

	GDT		RAM		LDT	
53=es->	50	----	A0000	-----	50	
	48		90000	-----	48	
	40		80000	-----	40	
	38		70000	-----	38	<- ds=3F
	30		60000	-----	30	<- ss=37
ldtr ->	28	-	50000	-----	28	
	20		40000	-----	20	
	18		30000	-----	18	
tr ->	10		20000	-----	10	
	08		10000	-----	08	<- cs=0F
gdtr->	** 00	----->	invalid	----->	00	

\*\* useless, null selector

**Note:** The digits within the tables are the offsets to each descriptor. The selector values ( CS=0F ) indicate which selector normally accesses the descriptor.

**Note:** Any selector containing the value 0-3 is the NULL selector which DOES NOT specify the first entry in the GDT. It is a place holder when a selector does not specify a descriptor. Any attempt to use the null selector results in a general protection exception.

**Note:** The descriptors in the LDT are 16-bit descriptors. This is one of the reasons that 16-bit programs still execute and fail in exactly the same manner as on previous versions of OS/2.

---

## Stacks

This section describes how most OS/2 programs use the stack.

Understanding the stack is generally straightforward. The stack is defined by the descriptor selected by the Stack Selector register or SS, and the stack pointer or SP. Stacks are always read/write. There are two basic operations on a stack, PUSH and POP. PUSH decrements the stack pointer and then stores the operand at the offset provided by SP in the stack segment. POP moves the data item at the offset provided by SP to the operand and then increments SP. SP ALWAYS POINTS TO THE LAST ITEM PUSHED. Stacks grow downward from higher addresses to lower addresses.

---

## Near CALL & RETurn

The near CALL instruction is used to invoke a subroutine. The instruction first pushes IP into the stack and then updates IP so that it contains the offset of the first instruction in the subroutine.

The near form of the RETurn instruction is really just a POP IP, which restores the saved content of IP. Execution continues at the instruction following the CALL.

---

## Far CALL & RETurn

The far CALL instruction is used to invoke a subroutine. The instruction first pushes CS into the stack, and then pushes IP. Next, it updates CS & IP so that they contain the selector:offset of the first instruction in the subroutine.

The far form of the RETurn instruction first pops IP, which restores the saved content of IP, and then pops CS, restoring it as well. Execution continues at the instruction following the CALL.

---

## Passing Parameters

Parameters are generally passed to a subroutine by putting them on the stack with PUSH instructions prior to the CALL. Parameters are removed from the stack in one of two ways:

By the caller ( 'C' convention ), generally by adding a constant to SP.

By the subroutine ( PASCAL convention ), by specifying the operand for the RETurn which is added to SP after the return address is POP'ed.

**Note:** 'C' convention PUSHes parameters from right to left.

**Note:** PASCAL convention PUSHes parameters left to right.

Because the NEAR versions of jump ( JMP ), CALL and RETurn DO NOT touch CS, there can be no change of privilege level during execution of any of them. The FAR versions of them do provide a new value for CS. If the new CS is the same privilege level as the current



level, the only change from above is that CALL PUSHes CS before PUSHing IP. RETurn POPs IP before POPing CS.

## Receiving Parameters

There is a register which can be used by a subroutine to access parameters very efficiently. This register is the Base Pointer. When it is used to obtain an offset, the default segment is the STACK SEGMENT. If the entry to a subroutine begins with these instructions the stack will look like the picture on the next page.

```
PUSH      BP
MOV       BP,SP
SUB       SP,sizeof( LOCAL DATA ITEMS )
```

This sequence is so common that there is a single instruction equivalent:

```
ENTER     sizeof( LOCAL DATA ITEMS ), 0
```

This allows all parameters to be accessed as BP plus the appropriate offset and local data elements to be accessed as BP minus the appropriate offset.

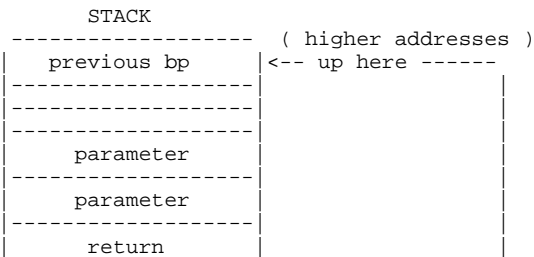
The instructions to exit are either:

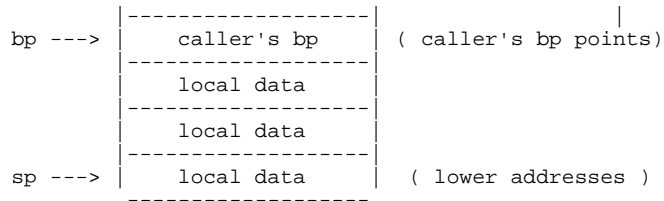
```
MOV       SP,BP
POP       BP
RET
or:
LEAVE
RET
```

## Why Do We Care About the Pascal Convention?

The Pascal convention was used by OS/2 1.x for those calls which access system functions which are implemented in a higher privilege level (ring) than the application. It is also used to call 16 bit Window Procedures. Two examples are DosAllocSeg and DosRead. The decision was made to use the Pascal convention because of the way the hardware protects access to instructions and storage which is more privileged. This type of interface, including hardware operation, is discussed in detail after basic stack operation has been discussed.

## Single Stack Frame





**Note:** A stack grows downward ( expand down ).

When this convention is followed the stack can be viewed as a series of 'stack frames'. Each stack frame has parameters and local data for some routine and linkage to the 'stack frames' used by the caller of that routine, etc... The saved BP values create a linked list in the stack segment which has all the information about each call including the return address. The process of following the chain back is referred to as 'unwinding the stack' and is an important aid to diagnosis when working on a problem.

## An Example of Using the Stack

This is a trivial example of how to pass and receive parameters, which is used to document where the stack pointer and base pointer are at the end of each instruction.

The example is 32-bit non-optimized code.

The subroutine, SUB, is designed to return the difference obtained by subtracting the second parameter from the first.

First, the relevant C code:

```
( main )          ( sub )
.                .
z=sub(A,B);        int sub(int x, int y)
.                {
.                return x-y;
.                }
```

Next, the assembler code

```
.          ( i ) initial condition
PUSH B      ; (01)          SUB:  PUSH EBP          ; (04)
PUSH A      ; (02)          MOV  EBP,ESP        ; (05)
CALL SUB    ; (03)          SUB  ESP,nn          ; (06)
ADD ESP,8    ; (12)          .
MOV Z,EAX    ; ( f ) final condition      . ( NOTE )
.
.
MOV EAX,[EBP+8] ; (07)
SUB EAX,[EBP+12] ; (08)
.
MOV ESP,EBP      (09)
POP EBP          (10)
RET              (11)
```

**Note:** At this point, the stack frame is established. If another, lower-level routine is called, the code to do so will look like the code seen in main, and a new stack frame will be established by that routine as soon as it receives control.

The new frame will be just below the current one.

## Stack Example

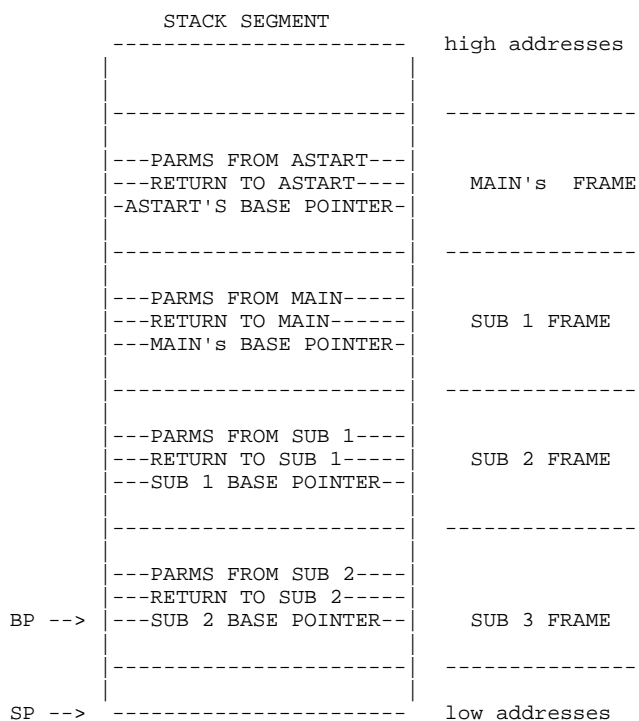
This example shows a stack, with ESP on the left, and EBP on the right.

```
ESP                      EBP
( higher addresses )      ( i,10 )
```

( i,12 )		?		
( 1 )		B		
( 2,11 )		A		
( 3,10 )		address of ADD		
( 4,9 )		EBP ( used by main )		( 5 )
.		automatic		
nn bytes		data		
.		for 'sub'		
( 6 )		( if used )		
				<-- where parameters for the next subroutine are put
		( lower addresses )		

**Note:** The numbers in parentheses indicate where the register points immediately after the numbered instruction on the previous page completes.

## Multiple Stack Frames



## A Stack From a Dump

```
DW SS:BP L10
001f:000014e6 1550 00f1 000f 02e8 18ae 0000 0000 0000
001f:000014f6 0000 0000 0000 0000 0000 0000 0000 0000
```

The first word, which is addressed by the current value in the BP register, is the near address of the next stack frame, 1550.

The next two words are a far return address, with the offset to the left of the selector. The return is to address F:F1.

The words following the return address are the parameters, if any were passed. There is no direct way to tell from the stack how many parameters were passed, or expected. To see the next frame,

```
DW 1550 L 10
001f:00001550 0000 0300 000f 0001 1560 001f 156e 001f
001f:00001560 1568 001f 0000 0000 4544 4f4d 0000 15c6
```

In this stack frame, the BP chain pointer is zero. This usually means that you have found all of the frames on this stack.

The return address for this frame is F:300. The parameters seem to be an integer, 1, and three far addresses, 1F:1560, 1F:156e, and 1F:1568. A little further inspection shows that the third address 1F:1568 is pointed to by the first, which is highly unusual. Actually, this is the stack frame received by 'main'. Main's parameters are as follows:

1. an integer, which tells it how many strings were found on the command line
2. the far address of a list of addresses, each of which points to one of the strings
3. the far address of a second list of addresses, each of which points to an environment variable. This list is terminated with a NULL POINTER, a far address in which both the selector and offset are zero.

Let's look at them.  
1F:1560 has the address 1f:1568. Near addresses default to the last selector used, so we are not required to supply it every time.

```
DA 1568
#001f:00001568 DEMO
```

Right, the name of the program was the first string on the command line. The first parameter indicates that there is only one string.

Let's look at a few of the environment variables.

```
DW 156E L8
001f:0000156e 15c6 001f 15da 001f 1608 001f 161f 001f
```

This gets us four far addresses. To see them all with only one input line, use the semicolon as a command delimiter and type away.

```
DA 15C6;DA 15DA;DA1608;DA 161F
001f:000015c6 WP_OBJHANDLE=132739
001f:000015da AUTOSTART=PROGRAMS,TASKLIST,FOLDERS,LAUNCHPAD
001f:00001608 BOOKSHELF=C:\OS2\BOOK;
001f:0000161f COMSPEC=C:\OS2\CMD.EXE
```

Notice that the tools are not very particular about spaces in the commands.

Lastly, to see the local data for the failing routine,  
DW SS:SP BP-2  
001f:000014be 02e8 18ae 00e3 2910 0017 0060 0017 0000  
001f:000014ce 0004 0017 c8cf 0000 0030 c949 c85a c8cf  
001f:000014de 0002 0000 00e6 1488  
and now you have it, displayed above.

---

## Application Documentation

We will briefly discuss what files are optionally generated by most compilers, and how to tell the linker to create the map file. After an explanation of the contents, and why some of the numbers are what they are, we will answer some questions using various parts of the optional application documentation.

---

## The .MAP File

When you look at a 16-bit map file, you will discover that it may have at least three sections. A 32-bit map file can have at least 4.

1. The first section is built in the same sequence as the executable.
2. The second section contains a list of all external symbols, sorted by the name of the symbol.

This is particularly useful when a programmer wants to find where some particular variable or routine is located.

3. The third section contains a list of the same symbols, sorted by the location of the symbol.

This is particularly useful when you know where something is, and want to find out if it has an external name, or what routine encompasses the address of interest.

4. The fourth section of a 32-bit map file contains a list of locations where the compiled code for each input line begins. This can tell you almost immediately which line of code failed, once you know which program, and where within the program the failing instruction was located.
- 

## The .COD File

Many 16-bit compilers will produce a file similar to a .COD file, although it may have a different file extension. For example, MicroFocus Cobol can produce a .GRP file, which has the same organization as the .COD file.

The format of this file is that of a mixed listing.

The listing generally contains an input line, identified by line number, followed by the machine instructions generated by the compiler, with the address to the left, the hex instruction in the middle, and an assembler form of the instruction on the right. Some of these files will actually be accepted as is by an assembler, but most compilers document the fact that this is not a supported feature of the compiler.

Obviously, if you know the offset of some instruction, perhaps one that caused a failure, you can use this listing to identify which line of the input program caused the generation of the failing instruction.

---

## Exercise 4: Application Documentation

Some typical files associated with 16- and 32-bit applications

---

## A 16-bit Map File

Part 1: Same sequence as executable.

DEMO

Start	Length	Name	Class
0001:0000	00292H	DEMO_TEXT	CODE
0001:0292	02BE6H	_TEXT	CODE
0001:2E78	00000H	C_ETEXT	ENDCODE
0002:0000	02910H	FAR_BSS	FAR_BSS
0003:0000	00042H	NULL	BEGDATA
0003:0042	007D8H	_DATA	DATA
0003:081A	0000EH	CDATA	DATA
0003:0828	00000H	XIFB	DATA
0003:0828	00000H	XIF	DATA
0003:0828	00000H	XIFE	DATA
0003:0828	00000H	XIB	DATA
0003:0828	00000H	XI	DATA
0003:0828	00000H	XIE	DATA
0003:0828	00000H	XPB	DATA
0003:0828	00004H	XP	DATA
0003:082C	00000H	XPE	DATA
0003:082C	00000H	XCB	DATA
0003:082C	00000H	XC	DATA
0003:082C	00000H	XCE	DATA
0003:082C	00000H	XCFB	DATA
0003:082C	00000H	XCF	DATA
0003:082C	00000H	XCFE	DATA
0003:082C	00006H	CONST	CONST
0003:0832	00008H	HDR	MSG
0003:083A	000FAH	MSG	MSG
0003:0934	00002H	PAD	MSG
0003:0936	00001H	EPAD	MSG
0003:0938	00226H	_BSS	BSS
0003:0B5E	00000H	XOB	BSS
0003:0B5E	00000H	XO	BSS
0003:0B5E	00000H	XOE	BSS
0003:0B60	00000H	c_common	BSS
0003:0B60	00A00H	STACK	STACK

Origin Group  
0003:0 DGROUP

**Note:** The numbers to the left of the colon look like the selector part of a far address, because that is what they will become. The linker has no idea what selectors will be assigned by the loader, so it simply calls the first segment 1, the next segment 2, and so on.

**Note:** The loader actually builds a table that shows the relationship between the selector assigned and the segment number from the map.

Part 2: Sorted by the name of the symbol

Address	Publics by Name		
0000:0000	Imp	DOSALLOCSEG	(DOSCALLS.34)
0000:0000	Imp	DOSCHGFILEPTR	(DOSCALLS.58)
0000:0000	Imp	DOSEXIT	(DOSCALLS.5)
0000:0000	Imp	DOSGETDBCSEV	(NLS.4)
0000:0000	Imp	DOSGETMACHINEMODE	(DOSCALLS.49)
0000:0000	Imp	DOSGETVERSION	(DOSCALLS.92)
0000:0000	Imp	DOSQHANDTYPE	(DOSCALLS.77)
0000:0000	Imp	DOSREAD	(DOSCALLS.137)
0000:0000	Imp	DOSREALLOCSEG	(DOSCALLS.38)
0000:0000	Imp	DOSSETVEC	(DOSCALLS.89)
0000:0000	Imp	DOSWRITE	(DOSCALLS.138)
0003:06E6		STKHQQ	
0001:2D3E		_brkctl	
0003:0938		_edata	
0003:0B60		_end	
0003:069B		_environ	
0003:0662		_errno	
0001:057A		_exit	
0001:24E6		_fflush	
0001:03F0		_fgets	
0001:295C		_flushall	
0001:275C		_free	
0001:0000		_gen	

0001:2836		_isatty
0001:29A0		_lseek
0001:00E2		_main
0001:2771		_malloc
0001:285A		_memset
0002:0000		_prime
0001:0394		_printf
0001:2618		_read
0001:0492		_sscanf
0001:2E64		_stackavail
0001:2024		_strlen
0001:282C		_ultoa
0001:2576		_ungetc
0001:29DE		_write
0003:06E2		__aaltstkovr
0003:04D6		__abrkp
0003:00D6		__abrktb
0003:04D6		__abrktbe
0003:04D8		__acfinfo
0003:00CC		__acmdl

0000:9876	Abs	__acrtmsg	
0000:9876	Abs	__acrtused	
0000:D6D6	Abs	__aDBdoswp	
0003:06A6		__adbgsmsg	
0000:D6D6	Abs	__aDBused	
0003:00CE		__aenvseg	
0003:00D4		__aexit_rtn	
0001:2E58		__aFlshl	
0001:28A2		__aFNalshl	
0000:0000	Imp	__AHINCR	(DOSCALLS.136)
0001:2BDD		__amalloc	
0001:2D1C		__amallocbrk	
0003:0816		__amblksiz	
0001:2CC0		__amexpand	
0001:2CFA		__amlink	
0001:0310		__amsg_exit	
0003:0042		__anullsize	
0003:0810		__asegl	
0003:0806		__asegds	
0003:04E6		__aseghi	
0003:04E8		__aseglo	
0003:0812		__asegn	
0003:0814		__asegr	
0003:00D0		__asizds	
0003:0702		__asizeC	
0003:0703		__asized	
0001:02A2		__astart	
0003:00D2		__atopsp	
0002:2710		__bufin	
0003:06EA		__cfltcvt_tab	
0003:06E8		__cflush	
0003:06A3		__child	
0001:2244		__chkstk	
0001:04F0		__cinit	
0001:0306		__cintDIV	
0001:2DF4		__cltoasub	
0001:05CA		__ctermsub	
0003:0704		__ctype	
0003:0704		__ctype_	
0001:2E01		__cxtoa	
0003:0669		__doserrno	
0003:0668		__dosmode	
0001:291F		__dosret	
0001:2910		__dosretf	
0003:0666		__dosvermajor	
0003:0667		__dosverminor	

0001:0591		__exit
0003:065A		__fac
0001:275C		__ffree
0001:05EC		__FF_MSGBANNER
0001:0702		__filbuf
0001:22D0		__flsbuf

0001:2771	__fmalloc
0003:081C	__fpinit
0001:223E	__fptrap
0001:08E0	__ftbuf
0001:2458	__getbuf
0001:098C	__input
0003:04EE	__iob
0003:05DE	__iob2
0003:0656	__lastiob
0003:066B	__nfile
0001:2B82	__nfree
0001:2B94	__nmalloc
0001:069C	__NMSG_TEXT
0001:06CC	__NMSG_WRITE
0001:2268	__nullcheck
0003:0669	__oserr
0003:066D	__osfile
0003:0666	__osmajor
0003:0667	__osminor
0003:0668	__osmode
0003:0666	__osversion
0001:156A	__output
0003:069F	__pgmptr
0003:0681	__pipe
0001:203C	__setargv
0001:0610	__setenvp
0003:0700	__sigintoff
0003:06FE	__sigintseg
0001:07FE	__stbuf
0001:228E	__stdalloc
0003:06AE	__stdbuf
0003:0664	__umaskval
0003:04EC	__aDBrterr
0003:04EA	__aDBswpflg
0003:0695	__argc
0003:0697	__argv

### Part 3: Sorted by location in storage

Address		Publics by Value
0000:0000	Imp	DOSGETMACHINEMODE (DOSCALLS.49)
0000:0000	Imp	DOSGETVERSION (DOSCALLS.92)
0000:0000	Imp	DOSREAD (DOSCALLS.137)
0000:0000	Imp	__AHINCR (DOSCALLS.136)
0000:0000	Imp	DOSEXIT (DOSCALLS.5)
0000:0000	Imp	DOSALLOCSEG (DOSCALLS.34)
0000:0000	Imp	DOSREALLOCSEG (DOSCALLS.38)
0000:0000	Imp	DOSCHGFILEPTR (DOSCALLS.58)
0000:0000	Imp	DOSWRITE (DOSCALLS.138)
0000:0000	Imp	DOSSETVEC (DOSCALLS.89)
0000:0000	Imp	DOSQHANDTYPE (DOSCALLS.77)
0000:0000	Imp	DOSGETDBCSEV (NLS.4)
0000:9876	Abs	__acrtmsg
0000:9876	Abs	__acrtused
0000:D6D6	Abs	__aDBdoswp
0000:D6D6	Abs	__aDBused
0001:0000		__gen
0001:00E2		__main
0001:02A2		__astart
0001:0306		__cintDIV
0001:0310		__amsmsg_exit
0001:0394		__printf
0001:03F0		__fgets
0001:0492		__sscanf
0001:04F0		__cinit
0001:057A		__exit
0001:0591		__exit
0001:05CA		__ctermsub
0001:05EC		__FF_MSGBANNER
0001:0610		__setenvp
0001:069C		__NMSG_TEXT
0001:06CC		__NMSG_WRITE
0001:0702		__filbuf
0001:07FE		__stbuf
0001:08E0		__ftbuf
0001:098C		__input



0001:156A	__output
0001:2024	__strlen
0001:203C	__setargv
0001:223E	__fptrap
0001:2244	__chkstk
0001:2268	__nullcheck
0001:228E	__stdalloc
0001:22D0	__flsbuf
0001:2458	__getbuf
0001:24E6	__fflush
0001:2576	__ungetc
0001:2618	__read
0001:275C	__free
0001:275C	__ffree

0001:2771	__fmalloc
0001:2771	__malloc
0001:282C	__ultoa
0001:2836	__isatty
0001:285A	__memset
0001:28A2	__aFNalshl
0001:2910	__dosretf
0001:291F	__dosret
0001:295C	__flushall
0001:29A0	__lseek
0001:29DE	__write
0001:2B82	__nfree
0001:2B94	__nmalloc
0001:2BDD	__amalloc
0001:2CC0	__amexpand
0001:2CFA	__amlink
0001:2D1C	__amallocbrk
0001:2D3E	__brkctl
0001:2DF4	__cltoasub
0001:2E01	__cxtoa
0001:2E58	__aFlshl
0001:2E64	__stackavail
0002:0000	__prime
0002:2710	__bufin
0003:0042	__anullsize
0003:00CC	__acmdl
0003:00CE	__aenvseg
0003:00D0	__asizds
0003:00D2	__atopsp
0003:00D4	__aexit_rtn
0003:00D6	__abrktb
0003:04D6	__abrktbe
0003:04D6	__abrkp
0003:04D8	__acfinfo
0003:04E6	__aseghi
0003:04E8	__aseglo
0003:04EA	__aDBswpflg
0003:04EC	__aDBrterr

0003:04EE	__iob
0003:05DE	__iob2
0003:0656	__lastiob
0003:065A	__fac
0003:0662	__errno
0003:0664	__umaskval
0003:0666	__osmajor
0003:0666	__dosvermajor
0003:0666	__osversion
0003:0667	__osminor
0003:0667	__dosverminor
0003:0668	__osmode
0003:0668	__dosmode
0003:0669	__doserrno
0003:0669	__oserr
0003:066B	__nfile
0003:066D	__osfile
0003:0681	__pipe
0003:0695	__argc
0003:0697	__argv

```

0003:069B      _environ
0003:069F      __pmptr
0003:06A3      __child
0003:06A6      __adbgmsg
0003:06AE      __stdbuf
0003:06E2      __aaltstkovr
0003:06E6      STKHQQ
0003:06E8      __cflush
0003:06EA      __cfltcvt_tab
0003:06FE      __sigintseg
0003:0700      __sigintoff
0003:0702      __asizeC
0003:0703      __asizeD
0003:0704      __ctype
0003:0704      __ctype_
0003:0806      __asegds
0003:0810      __asegl
0003:0812      __asegn
0003:0814      __asegr
0003:0816      __amblksiz
0003:081C      __fpinit
0003:0938      _edata
0003:0B60      _end

```

Program entry point at 0001:02A2

## A 16-Bit Code File

```

;      Static Name Aliases
;
;      $S180_inbuf      EQU      inbuf
;      TITLE      DEMO.C
;      .286p
;      .287
DEMO_TEXT      SEGMENT      WORD PUBLIC 'CODE'
DEMO_TEXT      ENDS
_DATA          SEGMENT      WORD PUBLIC 'DATA'
_DATA          ENDS
CONST          SEGMENT      WORD PUBLIC 'CONST'
CONST          ENDS
_BSS           SEGMENT      WORD PUBLIC 'BSS'
_BSS           ENDS
DGROUP         GROUP        CONST, _BSS, _DATA
ASSUME         CS: DEMO_TEXT, DS: DGROUP, SS: DGROUP
EXTRN          __acrtused:ABS
EXTRN          _printf:FAR
EXTRN          _scanf:FAR
EXTRN          _fgetc:FAR
_BSS           SEGMENT
COMM NEAR      _prime: 2:      5000
_BSS           ENDS
EXTRN          __iob:BYTE
_DATA          SEGMENT
$SG188 DB      'there are %u primes less than 65536', 0aH, 00H
$SG191 DB      '%u', 00H
$SG195 DB      'Enter number to factor: ', 00H
$SG197 DB      '%u', 00H
$SG198 DB      'Unable to convert number. Please try again', 0aH, 00H
$SG207 DB      '%u is prime', 0aH, 00H
$SG208 DB      '%u=%u', 00H
$SG212 DB      '*%u', 00H
$SG213 DB      0aH, 00H
_DATA          ENDS
_BSS           SEGMENT
$S180_inbuf     DW 028H DUP (?)
_BSS           ENDS
CONST          SEGMENT
$T20004 DW SEG _prime

```

```

CONST      ENDS
DEMO_TEXT  SEGMENT
            ASSUME  CS: DEMO_TEXT
;|***
;|*** #include <stdio.h>

; Line 2
;|*** #define INBUFSIZE 80
;|*** #define NPRIME 5000
;|*** unsigned short prime[NPRIME];
;|***
;|*** int gen(void)
;|*** {
; Line 8
PUBLIC _gen
_gen
PROC FAR
*** 000000      c8 24 00 00      enter    WORD PTR 36,0
*** 000004      57              push     di
*** 000005      56              push     si
;
;   ix = -6
;   l  = -16
;   ll = -14
;   npr = -2
;   q  = -4
;   t  = -10
;   tp = -8
;   tt = -12
;|*** unsigned short ix,l=2,ll=25,npr=3,q,t,tp=2,tt;
; Line 9
*** 000006      c7 46 f0 02 00      mov      WORD PTR [bp-16],2      ;l
*** 00000b      c7 46 f2 19 00      mov      WORD PTR [bp-14],25     ;ll
*** 000010      c7 46 fe 03 00      mov      WORD PTR [bp-2],3      ;npr
*** 000015      c7 46 f8 02 00      mov      WORD PTR [bp-8],2      ;tp
;|*** prime[0]=2;
; Line 10
*** 00001a      8e 06 00 00      mov      es,$T20004
*** 00001e      26 c7 06 00 00 02 00  mov      WORD PTR es:_prime,2
;|*** prime[1]=3;
; Line 11
*** 000025      26 c7 06 02 00 03 00      mov      WORD PTR es:_prime+2,3
;|*** prime[2]=5;
; Line 12
*** 00002c      26 c7 06 04 00 05 00      mov      WORD PTR es:_prime+4,5
;|*** for ( t=7 ; t<65530 ; t+=tp )
; Line 13
*** 000033      c7 46 f6 07 00      mov      WORD PTR [bp-10],7      ;t
*** 000038      c7 46 e2 04 00      mov      WORD PTR [bp-30],OFFSET _prime+4
*** 00003d      c7 46 e4 00 00      mov      WORD PTR [bp-28],SEG _prime
*** 000042      c7 46 de 06 00      mov      WORD PTR [bp-34],OFFSET _prime+6
*** 000047      c7 46 e0 00 00      mov      WORD PTR [bp-32],SEG _prime
;|***      {
; Line 14
;|***      tp=6-tp;
; Line 15
*** 00004c      b8 06 00      mov      ax,6
*** 00004f      2b 46 f8      sub      ax,WORD PTR [bp-8]      ;tp
*** 000052      89 46 f8      mov      WORD PTR [bp-8],ax      ;tp

;|***      if ( ll<=t )
; Line 16
*** 000055      8b 46 f6      mov      ax,WORD PTR [bp-10]      ;t
*** 000058      39 46 f2      cmp      WORD PTR [bp-14],ax      ;ll
*** 00005b      77 15      ja      $I170
;|***      {
; Line 17
;|***      l++;
; Line 18
*** 00005d      83 46 e2 02      add      WORD PTR [bp-30],2
*** 000061      ff 46 f0      inc      WORD PTR [bp-16]      ;l
;|***      ll=prime[l]*prime[l];
; Line 19
*** 000064      c4 5e e2      les      bx,DWORD PTR [bp-30]
*** 000067      26 8b 07      mov      ax,WORD PTR es:[bx]

```

```

*** 00006a      89 46 dc      mov     WORD PTR [bp-36],ax
*** 00006d      f7 e0      mul     ax
*** 00006f      89 46 f2      mov     WORD PTR [bp-14],ax      ;l1
;|***      }
; Line 20
;|***      for ( ix=2 ; ix<l ; ix++ )
; Line 21
*** 000072      be 02 00      $I170:      mov     si,2
*** 000075      39 76 f0      cmp     WORD PTR [bp-16],si      ;l
*** 000078      76 39      jbe     $FB173
*** 00007a      8b 46 f6      mov     ax,WORD PTR [bp-10]      ;t
*** 00007d      89 46 ec      mov     WORD PTR [bp-20],ax
*** 000080      c7 46 ee 00 00      mov     WORD PTR [bp-18],0
*** 000085      c7 46 e8 04 00      mov     WORD PTR [bp-24],OFFSET _prime+4
*** 00008a      c7 46 ea 00 00      mov     WORD PTR [bp-22],SEG _prime
*** 00008f      c4 7e e8      les     di,DWORD PTR [bp-24]
;|***      {
; Line 22
;|***      q=t/prime[ix];
; Line 23
*** 000092      26 8b 0d      mov     cx,WORD PTR es:[di]
*** 000095      8b 46 ec      mov     ax,WORD PTR [bp-20]
*** 000098      8b 56 ee      mov     dx,WORD PTR [bp-18]
*** 00009b      f7 f1      div     cx
*** 00009d      89 46 fc      mov     WORD PTR [bp-4],ax      ;q
;|***      tt=q*prime[ix];
; Line 24
*** 0000a0      f7 e1      mul     cx
*** 0000a2      89 46 f4      mov     WORD PTR [bp-12],ax      ;tt
;|***      if ( t==tt ) break;
; Line 25
*** 0000a5      39 46 f6      cmp     WORD PTR [bp-10],ax      ;t
*** 0000a8      74 09      je      $FB173
*** 0000aa      83 c7 02      add     di,2
*** 0000ad      46      inc     si
*** 0000ae      39 76 f0      cmp     WORD PTR [bp-16],si      ;l
*** 0000b1      77 df      ja      $L20000
;|***      }
;|***      if ( l==ix ) prime[npr++]=t;
; Line 27
*** 0000b3      39 76 f0      cmp     WORD PTR [bp-16],si      ;l
*** 0000b6      75 10      jne     $I175
*** 0000b8      c4 5e de      les     bx,DWORD PTR [bp-34]
*** 0000bb      8b 46 f6      mov     ax,WORD PTR [bp-10]      ;t
*** 0000be      26 89 07      mov     WORD PTR es:[bx],ax
*** 0000c1      83 46 de 02      add     WORD PTR [bp-34],2
*** 0000c5      ff 46 fe      inc     WORD PTR [bp-2] ;npr
;|***      }
; Line 28
*** 0000c8      8b 46 f8      $I175:      mov     ax,WORD PTR [bp-8]      ;tp
*** 0000cb      01 46 f6      add     WORD PTR [bp-10],ax      ;t
*** 0000ce      83 7e f6 fa      cmp     WORD PTR [bp-10],-6      ;t
*** 0000d2      73 03      jae     $JCC210
*** 0000d4      e9 75 ff      jmp     $L20002
*** 0000d7      89 76 fa      $JCC210:      mov     WORD PTR [bp-6],si      ;ix
;|***      return npr;
; Line 29
*** 0000da      8b 46 fe      mov     ax,WORD PTR [bp-2]      ;npr
*** 0000dd      5e      pop     si
*** 0000de      5f      pop     di
*** 0000df      c9      leave
*** 0000e0      cb      ret
*** 0000e1      90      nop
_gen      ENDP
;|***      }
;|***
;|*** int main(int argc, char *argv[])
;|***      {
; Line 33

```

```

PUBLIC _main
_main PROC FAR
*** 0000e2      c8 60 00 00      enter    WORD PTR 96,0
*** 0000e6      57              push     di
*** 0000e7      56              push     si
;   argc = 6
;   argv = 8
;   ix = -6
;   last = -10
;   nf = -8
;   fact = -76
;   input = -2
;   is = -12
;   q = -4
;|***   static char inbuf[INBUFSIZE];
;|***   int ix,last,nf;
;|***   unsigned short fact[32],input=0,is,q;
; Line 36
*** 0000e8      c7 46 fe 00 00      mov      WORD PTR [bp-2],0      ;input

;|***   last=gen();
; Line 37
*** 0000ed      0e              push     cs
*** 0000ee      e8 00 00          call     NEAR PTR _gen
*** 0000f1      89 46 f6          mov      WORD PTR [bp-10],ax      ;last
;|***   printf("there are %u primes less than 65536\n",last);
; Line 38
*** 0000f4      50              push     ax
*** 0000f5      1e              push     ds
*** 0000f6      68 00 00          push     OFFSET DGROUP:$SG188
*** 0000f9      9a 00 00 00 00      call     FAR PTR _printf
*** 0000fe      83 c4 06          add      sp,6
;|***   if ( 1<argc )
; Line 39
*** 000101      83 7e 06 01          cmp      WORD PTR [bp+6],1      ;argc
*** 000105      7e 25              jle      $I190
;|***   if ( 0==sscanf(argv[1],"%u",&input) ) argc=1;
; Line 40
*** 000107      8d 46 fe          lea      ax,WORD PTR [bp-2]      ;input
*** 00010a      16              push     ss
*** 00010b      50              push     ax
*** 00010c      1e              push     ds
*** 00010d      68 25 00          push     OFFSET DGROUP:$SG191
*** 000110      c4 5e 08          les      bx,DWORD PTR [bp+8]      ;argv
*** 000113      26 ff 77 06       push     WORD PTR es:[bx+6]
*** 000117      26 ff 77 04       push     WORD PTR es:[bx+4]
*** 00011b      9a 00 00 00 00      call     FAR PTR _sscanf
*** 000120      83 c4 0c          add      sp,12
*** 000123      0b c0            or       ax,ax
*** 000125      75 05            jne      $I190
*** 000127      c7 46 06 01 00     mov      WORD PTR [bp+6],1      ;argc
;|***   while ( 2>argc )
; Line 41
*** 00012c      83 7e 06 02          cmp      WORD PTR [bp+6],2      ;argc
*** 000130      7d 4b            jge      $FB194
*** 000132      8b 76 06          mov      si,WORD PTR [bp+6]      ;argc
***                                $L20005:
;|***   {
; Line 42
;|***   printf("Enter number to factor: ");
; Line 43
*** 000135      1e              push     ds
*** 000136      68 28 00          push     OFFSET DGROUP:$SG195
*** 000139      9a 00 00 00 00      call     FAR PTR _printf
*** 00013e      83 c4 04          add      sp,4
;|***   fgets(inbuf,INBUFSIZE,stdin);
; Line 44
*** 000141      1e              push     ds
*** 000142      68 00 00          push     OFFSET __iob
*** 000145      6a 50            push     80
*** 000147      1e              push     ds
*** 000148      68 00 00          push     OFFSET DGROUP:$S180_inbuf
*** 00014b      9a 00 00 00 00      call     FAR PTR _fgets
*** 000150      83 c4 0a          add      sp,10

```

```

;|***      if ( 0==sscanf(inbuf,"%u",&input) )
; Line 45
*** 000153      8d 46 fe      lea      ax,WORD PTR [bp-2]      ;input
*** 000156      16          push     ss
*** 000157      50          push     ax
*** 000158      1e          push     ds
*** 000159      68 41 00      push     OFFSET DGROUP:$SG197
*** 00015c      1e          push     ds
*** 00015d      68 00 00      push     OFFSET DGROUP:$S180_inbuf
*** 000160      9a 00 00 00 00 call     FAR PTR _sscanf
*** 000165      83 c4 0c      add      sp,12
*** 000168      0b c0      or       ax,ax
*** 00016a      75 11      jne      $FB194
;|***      printf("Unable to convert number. Please try again\n");
; Line 46
*** 00016c      1e          push     ds
*** 00016d      68 44 00      push     OFFSET DGROUP:$SG198
*** 000170      9a 00 00 00 00 call     FAR PTR _printf
*** 000175      83 c4 04      add      sp,4
;|***      else break;
;|***      }
; Line 48
*** 000178      83 fe 02      cmp      si,2
*** 00017b      7c b8      jl       $L20005
;|***      for ( ix=nf=0,is=input ; ix<last ; ix++ )
; Line 49
*** 00017d      2b c0      sub      ax,ax
*** 00017f      89 46 f8      mov      WORD PTR [bp-8],ax      ;nf
*** 000182      89 46 fa      mov      WORD PTR [bp-6],ax      ;ix
*** 000185      8b 46 fe      mov      ax,WORD PTR [bp-2]      ;input
*** 000188      89 46 f4      mov      WORD PTR [bp-12],ax     ;is
*** 00018b      83 7e f6 00      cmp      WORD PTR [bp-10],0      ;last
*** 00018f      7f 03      jg       $JCC399
*** 000191      e9 8e 00      jmp      $FB202
;|***      $JCC399:
*** 000194      8b 46 fa      mov      ax,WORD PTR [bp-6]      ;ix
*** 000197      d1 e0      shl      ax,1
*** 000199      05 00 00      add      ax,OFFSET _prime
*** 00019c      89 46 a6      mov      WORD PTR [bp-90],ax
*** 00019f      c7 46 a8 00 00 mov      WORD PTR [bp-88],SEG _prime
*** 0001a4      8d 46 b4      lea      ax,WORD PTR [bp-76]      ;fact
*** 0001a7      89 46 a2      mov      WORD PTR [bp-94],ax
*** 0001aa      8c 56 a4      mov      WORD PTR [bp-92],ss
*** 0001ad      8b 46 f6      mov      ax,WORD PTR [bp-10]     ;last
*** 0001b0      2b 46 fa      sub      ax,WORD PTR [bp-6]      ;ix
*** 0001b3      89 46 a0      mov      WORD PTR [bp-96],ax
*** 0001b6      01 46 fa      add      WORD PTR [bp-6],ax      ;ix
*** 0001b9      8b 4e fe      mov      cx,WORD PTR [bp-2]      ;input
;|***      {
; Line 50
;|***      q=input/prime[ix];
; Line 51
*** 0001bc      c4 5e a6      les      bx,DWORD PTR [bp-90]
*** 0001bf      26 8b 07      mov      ax,WORD PTR es:[bx]
*** 0001c2      89 46 aa      mov      WORD PTR [bp-86],ax
*** 0001c5      8b c1      mov      ax,cx
*** 0001c7      2b d2      sub      dx,dx
*** 0001c9      f7 76 aa      div      WORD PTR [bp-86]
*** 0001cc      89 46 fc      mov      WORD PTR [bp-4],ax      ;q
;|***      while ( q*prime[ix]==input )
; Line 52
*** 0001cf      8b 46 aa      mov      ax,WORD PTR [bp-86]
*** 0001d2      f7 66 fc      mul      WORD PTR [bp-4] ;q
*** 0001d5      3b c1      cmp      ax,cx
*** 0001d7      75 3d      jne      $FB205
*** 0001d9      26 8b 07      mov      ax,WORD PTR es:[bx]
*** 0001dc      89 46 b2      mov      WORD PTR [bp-78],ax
*** 0001df      8b f0      mov      si,ax
*** 0001e1      8b 46 a2      mov      ax,WORD PTR [bp-94]
*** 0001e4      8b 56 a4      mov      dx,WORD PTR [bp-92]
*** 0001e7      89 46 ac      mov      WORD PTR [bp-84],ax

```

```

*** 0001ea      89 56 ae      mov      WORD PTR [bp-82],dx
*** 0001ed      c4 7e ac      les      di,DWORD PTR [bp-84]
;|***          {
; Line 53
;|***          fact[nf++]=prime[ix];
; Line 54
*** 0001f0      26 89 35      mov      WORD PTR es:[di],si
*** 0001f3      83 c7 02      add      di,2
*** 0001f6      83 46 a2 02    add      WORD PTR [bp-94],2
*** 0001fa      ff 46 f8      inc      WORD PTR [bp-8] ;nf
;|***          input/=prime[ix];
; Line 55
*** 0001fd      8b c1      mov      ax,cx
*** 0001ff      2b d2      sub      dx,dx
*** 000201      f7 f6      div      si
*** 000203      8b c8      mov      cx,ax
;|***          q=input/prime[ix];
; Line 56
*** 000205      2b d2      sub      dx,dx
*** 000207      f7 f6      div      si
*** 000209      89 46 fc      mov      WORD PTR [bp-4],ax      ;q
;|***          }

; Line 57
*** 00020c      8b 46 b2      mov      ax,WORD PTR [bp-78]
*** 00020f      f7 66 fc      mul      WORD PTR [bp-4] ;q
*** 000212      3b c1      cmp      ax,cx
*** 000214      74 da      je      $L20006
;|***          }
; Line 58
*** 000216      83 46 a6 02    add      WORD PTR [bp-90],2
*** 00021a      ff 4e a0      dec      WORD PTR [bp-96]
*** 00021d      75 9d      jne      $L20008
*** 00021f      89 4e fe      mov      WORD PTR [bp-2],cx      ;input
;|***          }
;|***          if ( nf<2 ) return printf("%u is prime\n",is);
; Line 59
*** 000222      83 7e f8 02    cmp      WORD PTR [bp-8],2      ;nf
*** 000226      7d 14      jge      $I206
*** 000228      ff 76 f4      push     WORD PTR [bp-12]      ;is
*** 00022b      1e          push     ds
*** 00022c      68 70 00      push     OFFSET DGROUP:$SG207
*** 00022f      9a 00 00 00 00 call     FAR PTR _printf
*** 000234      83 c4 06      add      sp,6
*** 000237      5e          pop      si
*** 000238      5f          pop      di
*** 000239      c9          leave
*** 00023a      cb          ret
*** 00023b      90          nop
;|***          }
;|***          for ( ix=1 ; ix<nf ; ix++ )
; Line 61
*** 00024e      c7 46 fa 01 00 mov      WORD PTR [bp-6],1      ;ix
*** 000253      83 7e f8 01    cmp      WORD PTR [bp-8],1      ;nf
*** 000257      7e 29      jle      $FB211
*** 000259      8d 46 b6      lea      ax,WORD PTR [bp-74]
*** 00025c      89 46 a2      mov      WORD PTR [bp-94],ax
*** 00025f      8c 56 a4      mov      WORD PTR [bp-92],ss
*** 000262      8b 76 f8      mov      si,WORD PTR [bp-8]      ;nf
*** 000265      4e          dec      si
*** 000266      01 76 fa      add      WORD PTR [bp-6],si      ;ix
;|***          printf("%u",fact[ix]);
; Line 62

```

```

*** 000269      c4 5e a2          les     bx,DWORD PTR [bp-94]
*** 00026c      26 ff 37          push    WORD PTR es:[bx]
*** 00026f      1e              push    ds
*** 000270      68 83 00          push    OFFSET DGROUP:$SG212
*** 000273      9a 00 00 00 00      call    FAR PTR _printf
*** 000278      83 c4 06          add     sp,6
*** 00027b      83 46 a2 02          add     WORD PTR [bp-94],2
*** 00027f      4e              dec     si
*** 000280      75 e7          jne     $L20010
                                   $FB211:
;|***  return printf("\n");
; Line 63
*** 000282      1e              push    ds
*** 000283      68 87 00          push    OFFSET DGROUP:$SG213
*** 000286      9a 00 00 00 00      call    FAR PTR _printf
*** 00028b      83 c4 04          add     sp,4
*** 00028e      5e              pop     si
*** 00028f      5f              pop     di
*** 000290      c9              leave
*** 000291      cb              ret

_main      ENDP
DEMO_TEXT      ENDS
END
;|***  }

```

## Questions

Please answer the following questions, using the preceeding listings:

- How large is segment 2 of DEMO.EXE? \_\_\_\_\_
- What is the segment:offset of the 'fgets' routine? \_\_\_\_\_
- What is the segment:offset of the symbol 'fpinit'? \_\_\_\_\_
- Does DEMO.EXE call DosOpen? \_\_\_\_\_ How can you tell? \_\_\_\_\_
- Which routine begins at address 0001:29DE? \_\_\_\_\_
- How long is the routine named 'strlen'? \_\_\_\_\_
- Which routine contains address 0001:186A? \_\_\_\_\_
- How far into the routine is the previous address? \_\_\_\_\_
- What is the program's entry point? \_\_\_\_\_
- What is the name of the routine which is the entry point? \_\_\_\_\_
- What instruction mnemonic is at offset 00C5 in DEMO.EXE? \_\_\_\_\_
- What variable is in AX when the return at 00E0 executes? \_\_\_\_\_
- Which line in DEMO.C generated the above return? \_\_\_\_\_
- Offset 0188 in DEMO.C is in which C function? \_\_\_\_\_
- What variable name is used by the instruction at 0055? \_\_\_\_\_
- What is the purpose of the instruction at offset 0234? \_\_\_\_\_

**Note:** In the .cod file, the numbers in the assembler instructions are decimal.

The lines generated in the .cod file between offsets 00E7 and 00E8 Tell you where the local variables are stored, relatively speaking.



17. To what are the numbers like 8, -10, -76, -12 relative? \_\_\_\_\_
18. If a failure were to occur in routine 'gen', what command would you use to display only the variable 'npr'? \_\_\_\_\_ DW
19. How would you display the variable 't'? DW \_\_\_\_\_
20. Is the variable 'tp' in 'gen' at the same location as the variable 'nf' in main? Yes / No Explain.  
 \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_
21. Check the offsets for 'gen' and 'main' between the .map and the .cod files. Do they match? \_\_\_\_\_ Why/why not?  
 \_\_\_\_\_ Will the offsets always behave this way? Yes / No

Explain. \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

## A 32-bit Map File

DEMO

Start	Length	Name	Class
0001:00000000	000001A68H	CODE32	CODE 32-bit
0001:00001A68	000000030H	_MSGSEG32	CODE 32-bit
0002:00000000	00000006CH	DDE4_DATA32	DATA 32-bit
0003:00000000	00000005CH	DATA32	DATA 32-bit
0003:0000005C	0000000B0H	CONST32	CONST 32-bit
0003:0000010C	000000000H	BSS32	BSS 32-bit
0003:00000110	000002000H	STACK32	STACK 32-bit

Origin	Group
0000:0	FLAT
0003:0	DGROUP

Address	Publics by Name
0000:00000000	Imp DOS32FLATTOSSEL (DOSCALLS.425)
0001:00001A7A	DOS32GETMESSAGE
0001:00001A7A	Dos32GetMessage
0000:00000000	Imp DOS32IQUERYMESSAGECP (MSG.8)
0000:00000000	Imp DOS32SELTOFLAT (DOSCALLS.426)
0000:00000000	Imp DOS32TRUEGETMESSAGE (MSG.6)
0000:00000000	Imp DosAllocMem (DOSCALLS.299)
0000:00000000	Imp DosExit (DOSCALLS.234)
0000:00000000	Imp DosFreeMem (DOSCALLS.304)
0001:00001A7A	DosGetMessage
0001:00001A7A	DOSGETMESSAGE
0000:00000000	Imp DosWrite (DOSCALLS.282)
0001:00000F94	free
0001:00000000	gen
0001:000000AC	main
0001:000013F0	malloc
0001:00001A68	sig32
0002:00000008	_argc
0002:0000000C	_argv
0001:000001A4	_bufprint
0001:00001850	_DosFlatToSel
0001:00001848	_DosSelToFlat
0003:0000010C	_edata
0001:00000F48	_edcGetMessage
0003:00000110	_end
0002:00000004	_exeentry
0002:00000010	_have_freed
0001:00001010	_heapmin
0001:00001108	_heapmin_int
0001:000012E0	_ilog2
0001:0000162C	_MsgServ

```

0002:00000014      _pBucketArr
0001:000017FC      _PrintErrMsg
0001:0000017C      _printfieee
0001:00000154      _printf_ansi

```

```

0001:00001858      _setuparg
0001:00001304      _split_chunk
0001:00001A40      _terminate
0001:00001A50      _wfloatfmt
0001:000002BC      __dofmt
0001:000012E8      __isdigit

```

# Address                    Publics by Value

```

0000:00000000      Imp  DosFreeMem          (DOSCALLS.304)
0000:00000000      Imp  DOS32IQUERYMESSAGECP (MSG.8)
0000:00000000      Imp  DOS32SELTOFLAT      (DOSCALLS.426)
0000:00000000      Imp  DosAllocMem        (DOSCALLS.299)
0000:00000000      Imp  DOS32TRUEGETMESSAGE (MSG.6)
0000:00000000      Imp  DOS32FLATTOSEL     (DOSCALLS.425)
0000:00000000      Imp  DosWrite           (DOSCALLS.282)
0000:00000000      Imp  DosExit            (DOSCALLS.234)
0001:00000000      gen
0001:000000AC      main
0001:00000154      _printf_ansi
0001:0000017C      _printfieee
0001:000001A4      _bufprint
0001:000002BC      __dofmt
0001:00000F48      _edcGetMessage
0001:00000F94      free
0001:00001010      _heapmin
0001:00001108      _heapmin_int
0001:000012E0      _ilog2
0001:000012E8      __isdigit
0001:00001304      _split_chunk
0001:000013F0      malloc
0001:0000162C      _MsgServ
0001:000017FC      _PrintErrMsg
0001:00001848      _DosSelToFlat
0001:00001850      _DosFlatToSel
0001:00001858      _setuparg
0001:00001A40      _terminate
0001:00001A50      _wfloatfmt
0001:00001A68      sig32
0001:00001A7A      DOSGETMESSAGE
0001:00001A7A      Dos32GetMessage
0001:00001A7A      DOS32GETMESSAGE
0001:00001A7A      DosGetMessage
0002:00000004      _exeentry
0002:00000008      _argc
0002:0000000C      _argv
0002:00000010      _have_freed
0002:00000014      _pBucketArr
0003:0000010C      _edata
0003:00000110      _end

```

## Line numbers for DEMO.obj(DEMO.C) segment CODE32

Source Line Num	Src File Index	Flags (0X)	Seg:Offset (0X)
-----	-----	-----	-----
9	1	00	0001:00000000
11	1	00	0001:00000009
12	1	00	0001:0000001e
13	1	00	0001:00000024
14	1	00	0001:0000002b
15	1	00	0001:00000032
17	1	00	0001:00000040
18	1	00	0001:0000004b
20	1	00	0001:00000050
21	1	00	0001:00000051
23	1	00	0001:0000005a
27	1	00	0001:00000069
28	1	00	0001:00000082

29	1	00	0001:00000087
30	1	00	0001:00000097
31	1	00	0001:000000a4
34	1	00	0001:000000ac
39	1	00	0001:000000b2
40	1	00	0001:000000c8
41	1	00	0001:000000e5
42	1	00	0001:000000f4
43	1	00	0001:00000107
45	1	00	0001:00000114
46	1	00	0001:00000127
47	1	00	0001:00000144
48	1	00	0001:00000149

Record Number of Start of Source: 9  
Number of Primary Source Records: 26  
Source & Listing Files:  
File 1) DEMO.C  
File 2) C:\PMG\CSET\INCLUDE\os2.h  
File 3) C:\PMG\CSET\INCLUDE\os2def.h  
File 4) C:\PMG\CSET\INCLUDE\bse.h  
File 5) C:\PMG\CSET\INCLUDE\bsedos.h  
File 6) C:\PMG\CSET\INCLUDE\bsememf.h  
File 7) C:\PMG\CSET\INCLUDE\bsesub.h  
File 8) C:\PMG\CSET\INCLUDE\bseerr.h  
File 9) C:\PMG\CSET\INCLUDE\STDIO.H

Program entry point at 0001:00000F6C

## A 32-bit .ASM File, Produced by CSET/2

```

        TITLE    DEMO.C
        .386
        .387
        INCLUDELIB OS2386.LIB
        INCLUDELIB dde4nbs.lib
CODE32  SEGMENT DWORD USE32 PUBLIC 'CODE'
CODE32  ENDS
DATA32  SEGMENT DWORD USE32 PUBLIC 'DATA'
DATA32  ENDS
CONST32 SEGMENT DWORD USE32 PUBLIC 'CONST'
CONST32 ENDS
BSS32   SEGMENT DWORD USE32 PUBLIC 'BSS'
BSS32   ENDS
DGROUP  GROUP CONST32, BSS32, DATA32
        ASSUME  CS:FLAT, DS:FLAT, SS:FLAT, ES:FLAT
        EXTRN   DosAllocMem:PROC
        EXTRN   _printfiieee:PROC
        EXTRN   _DosFlatToSel:PROC
        EXTRN   _DosSelToFlat:PROC
        EXTRN   _exeentry:PROC
DATA32  SEGMENT
@STAT1  DB "non-zero return code fro"
DB "m DosAllocMem=%d",0aH,0H
        ALIGN  04H
@STAT2  DB "there are %u primes less"
DB " than 65536",0aH,0H
        ALIGN  04H
@STAT3  DB "%6u ",0H
@STAT4  DB 0aH,0H
        DD      _exeentry
DATA32  ENDS
BSS32   SEGMENT
BSS32   ENDS
CONST32 SEGMENT
CONST32 ENDS
CODE32  SEGMENT

```

```

;***** 9 int gen(int *prime)
        ALIGN 04H

gen      PUBLIC gen
        PROC
        PUSH     EBP
        MOV      EBP,ESP
        PUSH     EBX
        PUSH     ESI
        PUSH     EDI
        MOV      [EBP+08H],EAX;   prime

;***** 11      int ix,l=2,ll=25,npr=3,q,t,tp=2,tt;
        MOV      DWORD PTR [EBP-018H],019H;   ll
        MOV      DWORD PTR [EBP-014H],03H;     npr
        MOV      DWORD PTR [EBP-010H],02H;     tp

;***** 12      prime[0]=2;
        MOV      DWORD PTR [EAX],02H
;***** 13      prime[1]=3;
        MOV      DWORD PTR [EAX+04H],03H
;***** 14      prime[2]=5;
        MOV      DWORD PTR [EAX+08H],05H
;***** 15      for ( t=7 ; t<65530 ; t+=tp )
        MOV      ECX,[EBP-01CH]; ix
        MOV      EBX,07H
        MOV      EDI,02H
        ALIGN 04H
FELB6:

;***** 16      {
;***** 17      tp=6-tp;
        MOV      EDX,[EBP-010H]; tp
        NEG      EDX
        ADD      EDX,06H
        MOV      [EBP-010H],EDX; tp
;***** 18      if ( ll<=t )
        CMP      [EBP-018H],EBX; ll
        JG       FELB7
;***** 19      {
;***** 20      l++;
        INC      EDI
;***** 21      ll=prime[l]*prime[l];
        MOV      EDX,DWORD PTR [EAX+EDI*04H]
        IMUL     EDX,EDX
        MOV      [EBP-018H],EDX; ll
;***** 22      }
FELB7:

;***** 23      for ( ix=2 ; ix<l ; ix++ )
        MOV      ECX,02H
        CMP      EDI,02H
        JLE      FELB8
        ALIGN 04H
FELB9:
        MOV      [EBP-020H],EDI; @CBE17
        MOV      ESI,EAX
;***** 24      {
;***** 25      q=t/prime[ix];
;***** 26      tt=q*prime[ix];
;***** 27      if ( t==tt ) break;
        MOV      EDI,DWORD PTR [ESI+ECX*04H]
        MOV      EAX,EBX
        CDQ
        IDIV     EDI
        MOV      EDX,EDI
        MOV      EDI,[EBP-020H]; @CBE17
        XCHG     ESI,EAX
        IMUL     EDX,ESI
        CMP      EDX,EBX
        JE       FELB8

;***** 28      }
        INC      ECX

```

```

        CMP     ECX,EDI
        JL      FELB9
FELB8:

;***** 29      if ( l==ix ) prime[npr++]=t;
        CMP     EDI,ECX
        JNE     FELB12
        MOV     EDX,[EBP-014H]; npr
        MOV     DWORD PTR [EAX+EDX*04H],EBX
        INC     EDX
        MOV     [EBP-014H],EDX; npr
FELB12:
        MOV     EDX,EBX

;***** 30      }
        MOV     EBX,[EBP-010H]; tp
        ADD     EBX,EDX
        CMP     EBX,0fffaH
        JL      FELB6

;***** 31      return npr;
        MOV     EAX,[EBP-014H]; npr
        POP     EDI
        POP     ESI
        POP     EBX
        LEAVE
        RET
gen      ENDP

;***** 34 int main(int argc, char *argv[])
        ALIGN 04H

        PUBLIC main
main     PROC
        PUSH     EBX
        PUSH     ESI
        PUSH     EDI
        SUB      ESP,0cH

;***** 39      rc=DosAllocMem(&mem,16384,PAG_READ+PAG_WRITE+PAG_COMMIT);
        PUSH     013H
        PUSH     04000H
        LEA      ECX,[ESP+010H]; mem
        PUSH     ECX
        MOV      AL,03H
        CALL     DosAllocMem
        ADD      ESP,0cH

;***** 40      if ( rc ) return printf("non-zero return code from DosAllocMem=%d\n",rc);
        OR       EAX,EAX
        JE       FELB18
        PUSH     EAX
        MOV      EAX,OFFSET FLAT: @STAT1
        SUB      ESP,04H
        CALL     _printfieeee
        ADD      ESP,014H
        POP      EDI
        POP      ESI
        POP      EBX
        RET
FELB18:

;***** 41      last=gen(p=mem);
        MOV      EAX,[ESP+08H]; mem
        MOV      [ESP+04H],EAX; p
        CALL     gen
        MOV      ESI,EAX

;***** 42      printf("there are %u primes less than 65536\n",last);
        PUSH     ESI
        MOV      EAX,OFFSET FLAT: @STAT2
        SUB      ESP,04H
        CALL     _printfieeee
        MOV      EAX,ESI
        ADD      ESP,08H

```

```

;***** 43   for ( ix=0 ; ix<last ; ix++ )
            OR     EAX,EAX
            JLE     FELB20
            MOV     EBX,EAX
            MOV     EDI,[ESP+04H];  p
            XOR     ESI,ESI
            ALIGN  04H

FELB21:

;***** 44   {
;***** 45   printf("%6u ",p[ix]);
            PUSH    DWORD PTR [EDI+ESI*04H]
            MOV     EAX,OFFSET FLAT: @STAT3
            SUB     ESP,04H
            CALL    _printfieee
            ADD     ESP,08H

;***** 46   if ( 9==(ix%10) ) printf("\n");
            MOV     EAX,ESI
            MOV     ECX,0aH
            CDQ
            IDIV    ECX
            CMP     EDX,09H
            JNE     FELB22
            MOV     EAX,OFFSET FLAT: @STAT4
            CALL    _printfieee
FELB22:

;***** 47   }
            INC     ESI
            CMP     ESI,EBX
            JL      FELB21
FELB20:

;***** 48   return 0;
            XOR     EAX,EAX
            ADD     ESP,0cH
            POP     EDI
            POP     ESI
            POP     EBX
            RET
main        ENDP
CODE32     ENDS
END

```

## Questions

Please answer the following questions, using the preceeding listings:

1. How large is segment 1 of DEMO.EXE? \_\_\_\_\_
2. What is the segment:offset of the 'bufprint' routine? \_\_\_\_\_
3. What is the segment:offset of the symbol 'have\_freed'? \_\_\_\_\_
4. Does DEMO.EXE call DosWrite? \_\_\_\_\_ How can you tell? \_\_\_\_\_
5. Which routine begins at address 0001:12E8? \_\_\_\_\_
6. How long is the routine named 'terminate'? \_\_\_\_\_
7. Which routine contains address 0001:1888? \_\_\_\_\_
8. What is the program's entry point? \_\_\_\_\_

9. What is the name of the routine which has the entry? \_\_\_\_\_
10. How far into this routine is the actual entry point? \_\_\_\_\_
11. What is the first instruction mnemonic generated by line 28? \_\_\_\_\_
12. What variable is in EAX when the return at line 31 executes? \_\_\_\_\_
13. Offset 0124 in DEMO.C is in which C function? \_\_\_\_\_
14. What variable name is used by the instruction at 0040? \_\_\_\_\_
15. Where does the code for line 34 start? \_\_\_\_\_

**Note:** In the .asm file, the numbers in the assembler instructions are hex. You can tell because they are suffixed with 'H'.

The assembler code generated has the variable name following each line where it is referenced. This makes it easy to locate the variables, because you simply use the address expression in the instruction.

16. Look at line 11. To what are the numbers -18, -14, -10 relative?

- 
17. Look at the code generated for line 15.

Where will the variable 't' be found? \_\_\_\_\_

18. If a failure were to occur in routine 'gen', what command would you use to display only the variable 'npr'? DD \_\_\_\_\_
19. How would you display the variable 't'? DD \_\_\_\_\_
20. Is the variable 'l' in 'gen' at the same location as the variable 'ix' in main? Yes / No Explain.  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

-----

## Exercise 5: Unwinding a 16-bit Stack

Objectives:

1. To learn how to 'unwind' a stack. This is how to find the calling hierarchy which existed at some particular point, such as at the point of failure.
2. To learn how to 'mine' information from the stack frames.

Normally, every routine which has not returned to its caller will have a stack frame. Each stack frame normally contains the parameters passed to a routine, the return address for the routine, and the data which is local for that routine.

Start the dump formatter just as before, on the same dump.

Questions:

1. The convention states that BP or EBP will point to the current stack frame. SP will point to the lowest address which is in use.

Therefore, note the initial values for SP \_\_\_\_\_ and BP \_\_\_\_\_. Since SS is the selector that defines the stack, note which it is. Some analysts also note the limit of the SS descriptor, because that value bounds the range of both SP and BP.

SS \_\_\_\_\_ SSLIM \_\_\_\_\_

2. Display the current stack frame using DW SS:BP. This will show you only part of the frame, but most analysts do this because it makes following the chain so easy.

The first word is the offset, or near address, of the next frame. The second word is the offset part of the return address. If the call was a far call, the return must also be a far call. If this is the case, the third word is the selector part of the return address.

next stack frame \_\_\_\_\_ return offset \_\_\_\_\_ selector \_\_\_\_\_

3. Some number of words following the return address are the parameters passed. There is no way to know for certain how many parameters there are, unless you know how both the caller and the routine are written. Analysts typically write down a few words, as convenient.

parameter word# 1 \_\_\_\_\_ 2 \_\_\_\_\_ 3 \_\_\_\_\_ 4 \_\_\_\_\_

4. At this point we have gleaned what we can from this frame. Now you need to repeat the process for the rest of the stack frames.

Many analysts will follow the entire chain of stack frames before going to the system or application documentation to find the names of the routines involved, and the line numbers. Others choose to go back and forth, and put in the routine names and line numbers for each frame as they go.

The application documentation will tell you where variables are stored. Remember that each routine uses its own stack frame, so be certain to use the numeric value rather than the register name 'BP' to look at local data for routines other than the failing one.

If you display from SP to BP-2, or ESP to EBP-4, you will see the entire local data for the routine using the current stack frame. This can be quite nice for locating the individual variables.

Find the routine which failed by looking at the .MAP file.

Find the line number that failed by looking next at the .COD file.

The following variables are involved in the failure: 'npr' and 't'. their locations can be found in the .COD file.

Find the location of npr, \_\_\_\_\_ then display its value \_\_\_\_\_

Find the location of t, \_\_\_\_\_ then display its value \_\_\_\_\_

You may want to look at the call to the failing routine, before going away to find the programmer.

-----

## Exercise 6: Unwinding a 32-bit Stack

Objectives:

1. To learn how to 'unwind' a stack. This is how to find the calling hierarchy which existed at the point of failure.
2. To learn how to 'mine' information from the stack frames.

Normally, every routine which has not returned to its caller will have a stack frame. Each stack frame normally contains the parameters passed to a routine, the return address for the routine, and the data which is local for that routine.

Start the dump formatter by typing DF\_RET ..\DUMPS.162\DUMP04.DMP

Questions:

1. The convention states that BP or EBP will point to the current stack frame. ESP will point to the lowest address which is in use.

Therefore, note the initial values for ESP \_\_\_\_\_ and EBP \_\_\_\_\_. Since SS is the selector that defines the stack, note which it is.

SS \_\_\_\_\_ SSLIM \_\_\_\_\_ (not generally useful when SS is 53)

2. Display the current stack frame using DD SS:EBP. This will show you only part of the frame, but most analysts do this because it makes following the chain so easy.

The first doubleword is the offset, or near address, of the next frame. The second doubleword is the offset part of the return address. If the call was a far call, the return must also be a far call. If this is the case, the third doubleword is the selector part of the return address. IT IS RARE FOR 32-BIT PROGRAMS TO USE FAR ADDRESSES.

next stack frame \_\_\_\_\_ return offset \_\_\_\_\_

3. Some number of doublewords following the return address are the parameters passed. There is no way to know for certain how many parameters there are, unless you know how both the caller and the routine are written. Analysts typically write down a few doublewords, as convenient.



parameter doubleword# 1 \_\_\_\_\_ 2 \_\_\_\_\_ 3 \_\_\_\_\_ 4 \_\_\_\_\_

4. At this point we have gleaned what we can from this frame. Now you need to repeat the process for the rest of the stack frames.

```
eax=00080000 ebx=000097eb ecx=0000002d edx=00001000 esi=000000c5 edi=0000002d
eip=0001008e esp=000320c0 ebp=000320cc iopl=2 rf -- -- nv up ei pl zr na pe nc
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001a7000
005b:0001008e 891c90 mov dword ptr [eax+edx*27],ebx ds:00084000=invalid
```

```
DD SS:ESP EBP-4
0053:000320c0 00000000 00000000 00000000
```

```
DD SS:EBP L18
0053:000320cc 000320f8 000100f2 00080000 00080000
0053:000320dc 00080000 00000000 00000000 00000000
0053:000320ec 00010f8e 00000001 00070010 00000000
0053:000320fc 1bfbbf68 0000036d 00000000 00040000
0053:0003210c 0004030b 00000000 00000000 00000000
0053:0003211c 00000000 00000000 00000000 00000000
```

```
DD 330F8 L 10
0053:000320f8 00000000 1bfbbf68 0000036d 00000000
0053:00032108 00040000 0004030b 00000000 00000000
0053:00032118 00000000 00000000 00000000 00000000
0053:00032128 00000000 00000000 00000000 00000000
```

The first parameter passed by OS/2 is the load module handle.

```
.LMO 36d
hmte=036d pmte=%ff652c6c mflags=00903150 c:\pmg\classes\labs\lab4\demo.exe
obj vsize vbase flags ipagemap cpagemap hob sel
0001 00001a98 00010000 80002025 00000001 00000002 0361 000f r-x shr big
0002 0000006c 00020000 80002003 00000003 00000001 0000 0017 rw- big
0003 00002110 00030000 80002003 00000004 00000001 0000 001f rw- big
```

Wonder what the 00040000 and 0004030B are? Display them to see!

```
DB %40000
%00040000 57 50 5f 4f 42 4a 48 41-4e 44 4c 45 3d 31 33 32 WP_OBJHANDLE=132
%00040010 37 33 39 00 41 55 54 4f-53 54 41 52 54 3d 50 52 739·AUTOSTART=PR
%00040020 4f 47 52 41 4d 53 2c 54-41 53 4b 4c 49 53 54 2c OGRAMS,TASKLIST,
%00040030 46 4f 4c 44 45 52 53 2c-4c 41 55 4e 43 48 50 41 FOLDERS,LAUNCHPA
%00040040 44 00 42 4f 4f 4b 53 48-45 4c 46 3d 43 3a 5c 4f D·BOOKSHELF=C:\O
%00040050 53 32 5c 42 4f 4f 4b 3b-00 43 4f 4d 53 50 45 43 S2\BOOK;·COMSPEC
%00040060 3d 43 3a 5c 4f 53 32 5c-43 4d 44 2e 45 58 45 00 =C:\OS2\CMD·EXE·
%00040070 44 50 41 54 48 3d 43 3a-5c 50 4d 47 5c 4f 53 32 DPATH=C:\PMG\OS2
```

```
DB %4030B L 20
%0004030b 4c 41 42 34 5c 44 45 4d-4f 00 00 55 f0 8b c7 e8 LAB4\DEMO··Up·Gh
%0004031b d5 1d 01 00 ff 75 e8 e8-19 1e 01 00 83 c4 14 89 U·...uhh·...D·.
```

Frame at	Next Frame at	Return address	parameters:
_____	_____	_____	_____

Frame at	Next Frame at	Return address	parameters:
_____	_____	_____	_____

Frame at	Next Frame at	Return address	parameters:
_____	_____	_____	_____

Many analysts will follow the entire chain of stack frames before going to the system or application documentation to find the names of the routines involved, and the line numbers. Others choose to go back and forth, and put in the routine names and line numbers for each frame as they go.

The application documentation will tell you where variables are stored. Remember that each routine uses its own stack frame, so be certain to use the numeric value rather than the register name 'BP' to look at local data for routines other than the failing one.

If you display from ESP to EBP-2, or ESP to EBP-4, you will see the entire local data for the routine using the current stack frame. This can be quite nice for locating the individual variables.

Find the routine which failed by looking at the .MAP file.

Find the line number that failed by looking again at the .MAP file.

The following variables are involved in the failure: 'npr' and 't'. their locations can be found in the .ASM file.

Find the location of npr,\_\_\_\_\_ then display its value \_\_\_\_\_

Find the location of t,\_\_\_\_\_ then display its value \_\_\_\_\_ Hint: t has been optimized, and is in a register.

You may want to look at the call to the failing routine, before going away to find the programmer.

-----

## Requesting Kernel Services

If CALL targets a less privileged CS, or RET ( RETURN ) a more privileged CS, a general protection exception occurs by definition; a trusted program cannot directly invoke a less trusted one.

If CALL targets a more privileged CS, a general protection exception occurs because a less privileged program cannot access a more privileged object (code segment).

It is IMPOSSIBLE to DIRECTLY call a code segment which is a different privilege level than the caller.

It IS POSSIBLE to INDIRECTLY call a more privileged code segment.

-----

## The Task State Segment (TSS)

This hardware control block is used to control hardware multitasking, I/O access, and privilege transitions.

-----

## How to Find the TSS

There is a selector register named the task register (TR). This register has a GDT selector that chooses a descriptor whose type is TSS. This descriptor contains the base and limit for the TSS.

### Task State Segment Format

The fields from offset 4 to 1F are not changed by the hardware.

Offset(size)	Content	Offset(size)	Content
00(2)	link - previous tss selector		
04(4)	Ring 0 ESP	08(2)	Ring 0 SS
0C(4)	Ring 1 ESP	10(2)	Ring 1 SS
14(4)	Ring 2 ESP	18(2)	Ring 2 SS
1C(4)	CR3.	20(4)	EIP
24(4)	EFLAGS	28(4)	EAX
2C(4)	ECX	30(4)	EDX

34 (4)	EBX	38 (4)	ESP
3C (4)	EBP	40 (4)	ESI
44 (4)	EDI		
48 (2)	ES	4C (2)	CS
50 (2)	SS	54 (2)	DS
58 (2)	FS	5C (2)	GS
60 (2)	LDT selector	62 (2)	reserved
64 (2)	TFlags	66 (2)	IO Map

-----

## The Call Gate

An explanation of what a call gate provides, and how it works.

-----

## Why Have a Call Gate?

The CALL GATE is the mechanism by which an application requests services from the operating system. Integrity has several requirements which are not immediately obvious to most people.

1. The caller must be forced to use a designed entry point to prevent entry at an arbitrary location; for example, at a point after the parameters have been validated. This might cause the operating system to violate its own integrity or that of another application.
2. The parameters, as well as the rest of the stack, must be protected from the application while in use by the operating system to prevent changes by another thread in that application.
3. The return address must be protected from the application while the operating system is running to prevent other threads of the application from altering it in a way that would cause a return to the application in a privileged mode.

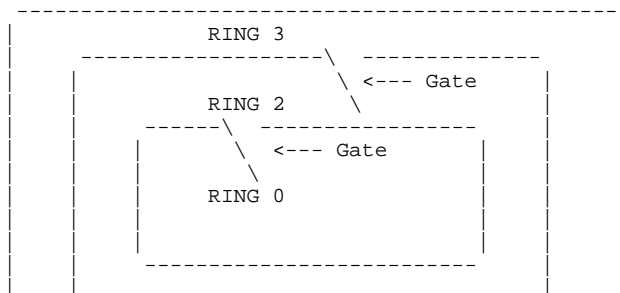
**Note:** A CALL GATE implements all of the above requirements.

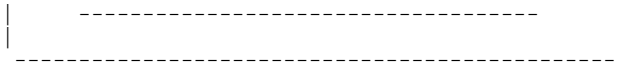
**Note:** A CALL GATE is a system descriptor which describes an entry point in a more privileged program which is accessible to less privileged programs.

-----

## Another View

A Gate is a 'service window' which describes the entry point of the gate and what size package is passed into the more protected ring.





-----

# Call Gate Contents

CALL GATE	
PL of Gate	Can I see this gate?
CS of entry	Where is the entry?
EIP of entry	Where is the entry?
Parm Count WC or DWC	What gets passed?
A Descriptor	

**Note:** Observe that the privilege level of the gate controls which privilege level programs can access the gate; the target privilege level is contained in the entry point CS value.

-----

# Call Gate Overview

When a FAR CALL contains a target code selector ( CS ) which is a CALL GATE, the processor ignores the offset ( IP ) contained in the instruction and gets the true target CS and offset ( IP ) from the CALL GATE. In addition, if the call is to a more privileged program, the processor locates a fresh stack for it to use, stores the current stack selector and stack pointer in the new, more privileged stack, copies the parameters from the old stack to the new one, and finally saves the return information in the new stack. All this happens during the execution of the call instruction.

Briefly, when the return occurs, all this gets undone.

-----

# Call Gate Detail

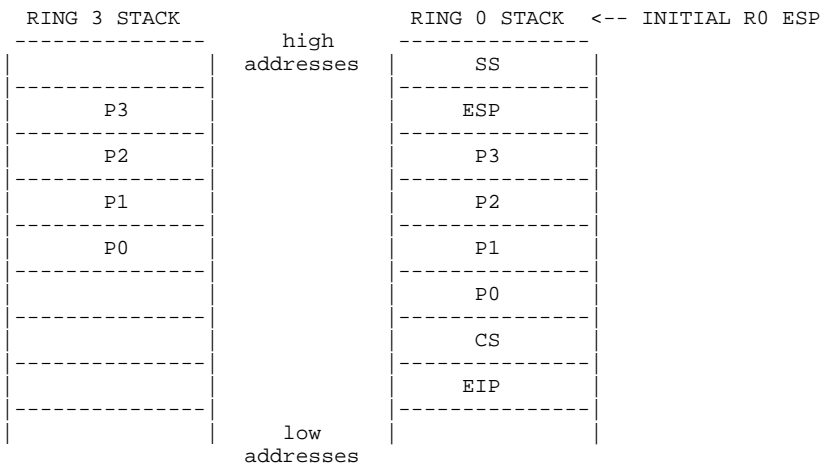
From less to more privileged, for example, Ring 3 to Ring 0

1. Verify new stack will hold linkage data. If not, stack fault, error code 0.
2. New SS, SP from TSS, based on PL of new CS.
3. Old SS:SP copied to new stack.
4. Parameters ( up to 15 words or doublewords ) copied from old to new stack.
5. Former CS:IP copied to new stack; SP now points here.
6. New CS, IP from Call Gate

-----

# A Ring Transition Picture

The following is how the stacks look at entry to the more privileged program:



**Note:** There is NO RETURN ADDRESS on the less privileged stack.

**Note:** The two items at the top of the more privileged stack are the less privileged SS and ESP.

**Note:** Subtract 8 from the SP value found in the TSS to find where the less privileged ESP and SS are stored. The values in the TSS are initial values, not the address of the first item pushed.

**Note:** A trap C in Ring 0 is usually a double fault.

When the processor detects a Stack Exception it needs to push an error code and a return address onto the stack of the exception handler. If this happens in Ring 0, there will be no privilege level transition, which includes switching to a new, protected stack. If the exception is due to stack growth, there is no place to push the error code or return address.

RESULT: TRAP 8

## Return Detail

From more to less privileged, for example, Ring 0 to Ring 3

1. Verify that all steps below will work. If not, general protection fault.
2. Pop IP, CS
3. Add immediate value to 'old' SP
4. Pop SP, SS
5. Add immediate value to 'new' SP
6. Zero every selector which has PL more privileged than the new CPL. This is required to maintain integrity because access validation is done only when a selector register is changed - not when it is used.

## Exercise 7: Looking at a Ring Transition

Objectives:

1. Introduction

To review previous knowledge of analysing traps

To begin getting familiar with the debug kernel

To learn how to identify the API targeted by a call gate

## 2. Techniques

To learn about the PATCH program

To learn about getting control when a module is loaded

## 3. Finding the TSS and the privileged stacks

To learn when you may need to find the TSS

To learn how to find the TSS

To learn how to find privileged stacks

## 4. Watching a ring transition

Look at the ring 0 stack before

Look at the ring 3 stack before

Actually execute a far call from ring 3 to ring 0.

Look at both stacks afterwards.

---

# Part 1: Introduction to the Debug Kernel

## Procedures: Introduction

1. Change to directory CLASS\LABS\LAB09
2. Execute OSPREY, see the failure, and the trap screen.

At the failure, record CS:EIP from the trap screen.

CS \_\_\_\_\_ EIP \_\_\_\_\_

At this point, it is too late to cause a dump. Dismiss the trap screen.

We will refer to the system on which the problem occurs at the Machine Under Test, or the MUT. The MUT is connected via a null modem cable to an adjacent machine, which we will call the debug terminal. Most of the debugging actions will occur from the debug terminal, on which we will run a public domain terminal emulation program, LOGICOMM. If you like LOGICOMM and intend to use it frequently, you should register it, which will also get you an improved version.

Let's use the debug kernel for the first time. First, we need to get its attention. The way to do this is to enter Control-C on the debug terminal, after starting LOGICOMM. The debug kernel defaults to settings  
9600, N, 8, 1

3. Start LOGICOMM on the debug terminal, then type Control-C.

The default response of the debug kernel is the registers at whatever point OS/2 was interrupted by the Control-C. This is not generally very useful. We need to get control where we want it, not at a random place.

4. Enter the command VSF\*

This tells the debug kernel that you want control on any interrupt which may be Fatal to a thread. The 'F' is for fatal, the '\*' is for 'any'.

Enter the command G (Go), so OS/2 can continue.

5. On the MUT, rerun OSPREY.

This time, you should get a group of lines on the debug terminal which tell you that a fatal failure has occurred.

Enter the command `DG CS` You will find that this is in ring 0.

Before we look at ring 0, let us find where ring 3 called ring 0, and also identify the API which was called.

Enter the command `.R` (the period is very significant!)

.R shows you the ring 3 registers, whereas R shows you the current ones.

CS=\_\_\_\_\_ EIP=\_\_\_\_\_ Does this match the trap screen?

```
eax=00000000 ebx=0000405c ecx=00000000 edx=00000001 esi=00000000 edi=000016b0
eip=00001bc3 esp=000011e4 ebp=0000120e iopl=2 -- -- -- nv up ei pl zr na pe nc
cs=000f ss=001f ds=001f es=001f fs=150b gs=0000 cr2=00000000 cr3=001a7000
000f:00001bc3 0bc0          or      ax,ax
```

We already know this instruction did not trap; the trap is in ring 0.

6. If we unassemble prior to 000f:1bc3, we will find a far call.

...1BBE call \_\_\_\_\_:0000

The instruction as hex data is \_\_\_\_\_

7. If you inquire about the descriptor by entering DG and the selector, You should see something similar to this

```
# DG 1xxx
1xxx CallG32 Sel:Off=0148:0000550a      DPL=3 P   DWC=7
```

Write down CS \_\_\_\_\_ EIP \_\_\_\_\_ DPL \_\_\_\_\_ DWC \_\_\_\_\_

If you enter the LN command with the values of CS and EIP from the call gate, you will identify the API which is called via this gate.

8. We know how to find parameters on the ring 3 stack, DW SS:SP

We can also find them on the ring 0 stack, but at this point, the kernel has already manipulated some of the addresses, so there is not an exact match. We need to get control at the point of the call at 1BBE.

9. Enter the command `GT` which will GoThrough the trap.

-----

## Part 2: Some Techniques

Procedures to get control at a point other than a trap:

One approach is to use clever breakpoints within OS/2. Stopping at the first executable instruction of a program

1. We will make use of a couple of breakpoint commands

This command tells the debug kernel that we want control on the debug terminal at some specific point. The problem is that the place where we would like to get control is not loaded into memory until we run the program, and it is difficult at best to type Control-C at just the right time.

2. The initial breakpoint uses the fact that almost all programs use the DOSCALL1 DLL, which appears to have instance initialization.

Enter the command `BP DosLibIDisp,'p'`

The content of the quoted string is the command to execute when we arrive at the breakpoint. This will assure us that we are in the correct context, because the output of 'p' includes the module name.

Let the MUT run, and execute OSPREY once again.

You will probably get control in the context of OSPREY. If not, issue 'g' again a time or two until you are.

3. At this point, OSPREY has been loaded, so we can set a breakpoint.

If you simply try the command BP 0F:1BBE, you will discover that the page is not yet loaded. There are two ways around this problem.

- a. Use a register breakpoint, BR E,0F:1BBE
- b. Cause OS/2 to bring the page in with .I 0F:1BBE

Then reenter the BP command from above.

4. However, this is 'cheating' because we already knew where to stop.

To find the address of the first instruction at this point, enter the command .M 0F:0 Find the MTE handle, hmte.

Issue the .LMO command with the HMTE as the parameter.

Alternatively, try .LMO 'OSPREY', which works sometimes.

The output of the .LMO command includes the linear address of the MTE.

Display the MTE as doublewords, and get the address of the SMTE from the output; it is in the second doubleword.

Display the SMTE as doublewords, and you can find the entry point in the second and third words displayed.

Now you can set a breakpoint at the entry to any module.

The PATCH program

1. On the MUT, execute the EXEHDR utility against OSPREY.EXE.

EXEHDR is distributed with the developers' toolkit.

The output will provide you information you need to patch a program successfully. The last part of the output should look like

```
Module:           OSPREY
Description:      OSPREY.EXE
Data:            NONSHARED
Initial CS:IP:    seg  1 offset 0088
Initial SS:SP:    seg  3 offset 0000
Extra stack allocation: 0a00 bytes
DGROUP:          seg  3
```

```
no. type address file mem  flags
  1 CODE 00000200 0247d 0247e
  2 DATA 00000000 00000 00200
  3 DATA 00002800 007cb 00960
```

There are two things we will need in this listing.

2. The entry point, or initial CS:IP is \_\_\_\_\_:
3. The location in the file where that segment begins \_\_\_\_\_

The columns labelled 'file' and 'mem' are the sizes of the segment in the file, and in memory. The difference is due to uninitialized data, which is not stored, saving space and reducing program load time.

To find the location of an instruction in the file, add the offset to the file address.

4. To get control, we will replace a byte with the hex value 'CC', which is a special one-byte instruction, Int 3, or BreakPoint.
5. We will patch the call instruction at 1BBE.

Add the offset, 1BBE to the file address 0200 \_\_\_\_\_

If you cannot add hex, get the debug kernel's attention, and then type in ? 1BBE+0200. ? is a general purpose evaluation command.

6. We now know where we want to patch the program. Let's do it.

On the MUT, enter the command PATCH OSPREY.EXE



The patch address was calculated above; enter it.

The byte you are about to replace is hex \_\_\_\_\_

Type `CC` then press enter, and complete the confirmations.

We have now patched the program.

7. Execute the program on the MUT; you get control at the INT 3.

We need to put back the hex data which was originally there, so as not to introduce another problem. We will use the enter command.

Type the command `E CS:IP`

You will see the 'CC', type the original data value and press enter.

Type the command `.R` and you should see the original far call.

8. This is one way to get control.

It has problems if the MUT is not where you can touch it.

Type the commands `G` then `GT` to let OSPREY finish.

9. Patch OSPREY back to its original content if you wish.

---

## Part 3: Finding the TSS

It is relatively simple to find and display this critical control block which is used by the hardware for ring transitions.

1. Get the debug kernel's attention, so you can display data.
2. The TSS is located via the Task Register (TR), which is a selector.

You can find the value in TR by entering `? TR`

Entering `RT` toggles register terse mode. Try `R` before and after entering `RT`. You can look for TR in the output.

You really do not want TR, but the TSS, which is at `TR:0`.

3. Enter

`DD TR:0` to display the TSS as doublewords

`DT TR:0` to format the TSS.

4. The first doubleword is the link field.

It indicates which TSS called this one through a task gate.

The next two doublewords are the ESP and SS for entry to ring 0.

The next pair of doublewords are unused by OS/2; they would have the ESP and SS for entry into ring 1.

The next pair of doublewords are the ESP and SS for entry to ring 2.

5. To display the stack used at entry to ring 0,

use the `DD` command with the SS and ESP values from the TSS; BUT Stacks grow downward, so put `-80` after the ESP value. 80 is the number of bytes displayed by default; this will show you the top of the stack for ring 0, with the saved SS value as the last item shown.

---

## Part 4: Watching a Ring Transition

We will watch a ring transition by stopping on an instruction which we know causes a ring transition, display both stacks, then single step the instruction, and look at both stacks again.

Get control in OSPREY so that the next instruction is at 0F:1BBE.

1. Display the call gate by using `DG` and the selector from the call.

Write down the target CS\_\_\_\_\_ EIP\_\_\_\_\_ DW\_\_\_\_\_ PL\_\_\_\_\_

2. Display the ring 3 stack as WORDS

so you can see as many DOUBLEWORDS as are passed through the gate.

Display the ring 0 stack as words, too. It is technically incorrect to do this, but for the purposes of this exercise, it makes things easy.

3. Use the command `T` to execute the call instruction.

Now, again display the ring 0 stack as words again.

4. Compare the ring 0 content now with the content of the ring 3 stack.

Do not overlook the ring 3 SS and ESP at the top of the ring 0 stack.

Do not overlook the return address in the ring 0 stack, following the parameters which were copied by the hardware as it executed the call.

5. TIMESAVER:

If you know what API will be called, you can simply set the breakpoint at the API, by using its name. A side effect is that every thread which calls the API will stop, so you may want to use something like `'p*'` as the command to execute at the breakpoint, which makes it easy to see when the thread of interest is there.

This lab is now complete. However, if you let it run to the failing instruction, you will find an additional detail about this API, namely that because only 13 words were pushed, and 7 doublewords are needed to get them all copied into the ring 0 stack, there is one more detail we can see, namely how the difference (two bytes) is handled.

If you display the ring 0 stack once again, it has been changed!

The return will need to add enough to the ring 0 stack pointer that it can find the ring 3 stack successfully; this is also what is added to the ring 3 stack pointer, because both stacks must be cleaned up. In order that this not be destructive of what is already on the ring 3 stack, the ring 0 entry code has adjusted the saved ring 3 ESP downward by 2 before the trap occurs. This is an example of some of the work that has been done within the ring 0 stack by the privileged code.

## Exercise 8: Identifying the Owner of Storage

Objectives:

1. To learn how to find out where a part of storage originated
2. To learn how to find out what module contains it, if not dynamically acquired

Every piece of storage has an owner. Storage owned by OS/2 may not have all the storage accounting information which is kept for storage used by applications. The most common clue that this situation has occurred is the presence of the `'UVirt'` flag (bit 52) in the descriptor. The next most common clue is that the procedure below may fail if complete storage accounting has not been done.

Within OS/2, handles are used extensively. Generally, a handle is nothing more than an index into some table or other. For diagnostic purposes, one can treat it as a 'magic number' that can be used as an operand on certain commands.

The initial objective is to find the module table entry which the loader built when the module was loaded. This will relate storage to the 'far' addresses in the link map.

The procedure is slightly different for private and shared storage.

With practice, one learns quickly what selectors are likely to be private, and which are likely to be shared. Refer to the address space picture which appears earlier, to refresh your memory about private and shared storage.

One way to tell is to display the entire LDT ( using 'DL' ), and to look for the gap between 'low numbered' and 'high numbered' selectors.

If the address is private, there will likely be many processes that define the address, and the data is likely different for each. You will need to find which process is the one you want.

The dump formatter command '.I' will show you not only the handle of the module table entry for the executable which caused this process to exist, but also will show you the handle of the 'PTDA', which is the key control block for a process.

The command usually used to identify storage is the '.M' command.

If issued with a shared address, the output has the handle of the module table entry. If issued for a private address, you get a set of output lines for every process which contains the address. In this case, you will need to use the hPTDA, or PTDA handle from the '.I' command to determine which set of output lines to use.

Once you have the handle of the module table entry, issue the command .LMO <handle>.

The command will not only give you the full path name of the module, but will also format a table which has a column (toward the right) titled 'sel'. This is the selector assigned. The first line of output is for the first segment in the link map, the second line is for the second segment, and so on. Thus, you can convert the selector:offset in the dump to a segment:offset in the correct link map.

```
.I
PROCESS slot:23 Pid:0008 Ord:0001
PTDA     handle=032e address=%ad6d97f0
MTE      handle=0363 address=%ff666d4c (DEMO)
SMTE     address=%fel4abe8
LDT      handle=035c address=%ac6d7000
CODE:    user (cs:eip)#000f:000000be cbargs=
STACKS:  user (ss:esp)#001f:000014be(active)
         ring2(ss:esp)#0036:00001000(bottom)
         ring0 tcbframe=%fe023f58 bottom=%fe023f9c

.M CS:IP
*har     par      cpg      va      flg next prev link hash hob   hal
01f5 %ff821b18 00000010 %00010000 1c9 01f6 01f3 00fa 0000 0131 0000 hptda=0240
00fa %ff820586 00000010 %00010000 1d9 0102 00f9 0000 0000 0131 0000 hptda=0117
hob     har hobnxt flgs own  hmte  sown,cnt lt st xf
0131 01f5 0000 0838 0132 0132 0000 00 00 00 00 shared      c:pmsHELL.exe

*har     par      cpg      va      flg next prev link hash hob   hal
0177 %ff821044 00000010 %00010000 179 0178 0175 0000 0000 01be 0000 hptda=01b9
hob     har hobnxt flgs own  hmte  sown,cnt lt st xf
01be 0177 0000 002c 01b9 01bf 0000 00 00 00 00 UNKNOWN

*har     par      cpg      va      flg next prev link hash hob   hal
01f5 %ff821b18 00000010 %00010000 1c9 01f6 01f3 00fa 0000 0131 0000 hptda=0240
00fa %ff820586 00000010 %00010000 1d9 0102 00f9 0000 0000 0131 0000 hptda=0117
hob     har hobnxt flgs own  hmte  sown,cnt lt st xf
0131 01f5 0000 0838 0132 0132 0000 00 00 00 00 shared      c:pmsHELL.exe

*har     par      cpg      va      flg next prev link hash hob   hal
02b5 %ff822b98 00000010 %00010000 1d9 02b6 02b2 0000 0000 0322 0000 hptda=031c
hob     har hobnxt flgs own  hmte  sown,cnt lt st xf
0322 02b5 0000 0838 0327 0327 0000 00 00 00 00 shared      c:cmd.exe

*har     par      cpg      va      flg next prev link hash hob   hal
02e6 %ff822fce 00000010 %00010000 1d9 02e7 02e5 0000 0000 0362 0000 hptda=032e
hob     har hobnxt flgs own  hmte  sown,cnt lt st xf
0362 02e6 0000 0838 0363 0363 0000 00 00 00 00 shared      c:demo.exe

.LMO 363
hmte=0363 pmte=%ff666d4c mflags=00803142 c:\pmg\pete\demo16\demo.exe
seg  sect psiz vsiz hob  sel  flags
0001 0001 2e78 2e78 0362 000f 2d20 code shr rel
0002 0000 0000 2910 0000 0017 0c01 data
0003 0019 0937 1560 0000 001f 0d01 data rel
```

Start the dump formatter by typing DF\_RET ..\DUMPS.162\DUMP01.DMP

Procedure:

1. Enter the command '.I'

The PTDA handle is \_\_\_\_\_, the module table entry handle is \_\_\_\_\_

2. Enter the command '.M CS:IP' to identify Memory at CS:IP.

Which 'har' line is for our process? har=\_\_\_\_\_

What is the hmte value from this set of lines? hmte=\_\_\_\_\_

**Note:** This is exactly what the '.l' command showed you, because this is what the .l command does internally.

3. Enter the command '.LMO ', followed by the hmte value.

What is the full path name of the module that contains CS:IP?

\_\_\_\_\_

4. What is the segment number which has been assigned selector 000F? \_\_\_\_\_

5. What address would you look for in the link map to find CS:IP?
- \_\_\_\_\_

6. Now, repeat the same steps using the data in the next few displays. The address of interest is DFDF:9324

7. What is the privilege level of this segment? \_\_\_\_\_

8. What is its size? \_\_\_\_\_

9. What is the command to identify memory at this address? \_\_\_\_\_

Issue it. The lines which start hco= are context records, which indicate all of the contexts (processes) that can reference this address. It is extremely likely to be a shared address.

10. Issue the '.LMO' command for the module table entry handle.

11. What module is this? Full path name is \_\_\_\_\_

12. Which segment in the module contains this address? \_\_\_\_\_

13. Therefore, in the .map file, the address will be \_\_\_\_\_:\_\_\_\_\_

```
DG DFDF
LDT
dfd  Code    Bas=1bfb0000 Lim=0000d4ef DPL=2 P  RE C  A

.M DFDF:9324
*har    par      cpq      va      flg next prev link hash hob   hal
00b7 %ff81ffc4 00000010 %1bfb0000 3d9 0075 00b8 0000 00b4 00c5 0000 hco=0026c
hob    har hobnxt flgs own  hmte  sown,cnt lt st xf
00c5 00b7 0000 0838 00bf 00bf 0000 00 00 00 00 shared    c:doscall11.dll
hco=026c pco=ffe62c37 hconext=00184 hptda=032e f=1c pid=
hco=0184 pco=ffe627af hconext=00014 hptda=031c f=1c pid=
hco=0014 pco=ffe6207f hconext=00089 hptda=0240 f=1c pid=
hco=0089 pco=ffe622c8 hconext=00021 hptda=01b9 f=1c pid=
hco=0021 pco=ffe620c0 hconext=00000 hptda=0117 f=1c pid=

.LMO BF
hmte=00bf pmte=%ff66ef3c mflags=0498b594 c:\os2\dll\doscall11.dll
obj  vsize  vbase  flags  ipagemap cpagemap hob  sel
0001 00000360 1bf80000 80009025 00000001 00000001 00c8 dfc6 r-x shr alias iopl
0002 0000aa30 1bf90000 80002025 00000002 0000000b 00c7 dfcf r-x shr big
0003 0000d519 1bfa0000 8000d025 0000000d 0000000e 00c6 dfd6 r-x shr alias conf iopl
0004 0000d4f0 1bfb0000 8000d025 0000001b 0000000e 00c5 dfde r-x shr alias conf iopl
0005 00001140 13f90000 80001023 00000029 00000002 00c4 9fcf rw- shr alias
0006 00001af0 13fa0000 80001003 0000002b 00000002 0000 9fd7 rw- alias
0007 00000e44 13fb0000 80001023 0000002d 00000001 00c2 9fdf rw- shr alias
0008 00000550 13fc0000 80001003 0000002e 00000001 0000 9fe7 rw- alias
0009 00001000 13fd0000 80001023 0000002f 00000000 00c0 9fef rw- shr alias
000a 00001000 13fe0000 80001023 0000002f 00000000 00be 9ff7 rw- shr alias
```

## Steps to Diagnose a Trap

The intent of the following material is to illustrate a proven method for finding the cause of a trap in an application program. By first learning how to solve the simplest problems, one will have a much better basis for approaching more difficult problems. Historically, problem solving skills have been largely self-taught. Much can be learned by observing others solve problems. Many problems can be solved quickly by using significant short-cuts and assumptions and then verifying them. When a novice observes an experienced diagnostician, the activities are difficult to understand, and may lead to the opinion that each problem has its own special method for solution, which in turn leads to questions about when to use which method.

The following process will lead you to the cause of a trap.

Remember to take notes as you proceed. This will help if you are interrupted, and want to continue later, or if you need to explain to someone else what you found, and what facts led you to a particular analysis of the situation. You can obviously do this manually, but you can use a log file more easily. Just type '?' followed by whatever you wish to log. The tools will evaluate the string, supplying the trailing quote, and show you the string, thus adding your thoughts to the log.

1. Locate the failing instruction.

If you cannot do this, you have no place to start. Most operating systems will provide at least an excellent clue to the location of the failing instruction, if not its exact address.

2. Determine why the failing instruction will not execute.

A knowledge of hardware operation, or a reference manual kept handy, is essential for this step. At the very worst, each of the possible exceptions described in the manual can be eliminated one by one until the cause is found.

Until you know why the instruction will not execute, you do not know what went wrong at the machine level. Conversely, as soon as you do know, you are prepared to begin the diagnosis of how things got into such a state. Observe that this does not require knowledge of C, FORTRAN, COBOL, SMALLTALK, etc. It requires only hardware knowledge.

3. Analyse how the conditions for failure occurred. It may be that an address calculation was done incorrectly, or that the failure was due to an invalid parameter. If the former, you now need only to discover what program has done this, and where in that program the error occurred. Skip the next two steps.
4. If an invalid parameter has been received, you must now update your notion of the cause of the problem. You need to consider the call as the location of the failure, and the specific parameter value as the reason why the called routine did not execute.
5. You must now analyse how the parameter was created, and where it came from. Unwind one stack frame, and return to step 3.
6. You now know what caused the problem, and now it is time to identify the failing program, locate the failing line, find the value of the program's variables, and, in general, collect all the data the programmer would have had if the failure had occurred at his desk. This step is usually a mechanical one.

Once this is done, go find the programmer, and turn over all you know about the problem. Be prepared to continue helping, or to show the programmer how to get additional information.

---

## The OS/2 System Trace

The System Trace facility is used to record a sequence of system events, function calls, or data in a fixed-size circular buffer as requested by calls to its API's. The buffer must be allocated during the processing of CONFIG.SYS.

If you have a TRACEBUF statement in CONFIG.SYS, a trace buffer is allocated, allowing you to use the TRACE command successfully later.

If any valid TRACE statements are in CONFIG.SYS (including TRACE=OFF), a trace buffer will be allocated for the default size of 4K, if TRACEBUF is not specified. This means that the statement TRACE=OFF will actually enable system tracing, which seems counter-intuitive to many people.

If you do not specify TRACE or TRACEBUF statements in CONFIG.SYS, OS/2 does not allocate a trace buffer and system tracing is disabled.

After the trace data is recorded, the trace formatter is used to retrieve the data from the system trace buffer and present it on your display, and optionally, to print it, or save it in a file.

---

# TRACEBUF and TRACEFMT

TRACEBUF=x

TRACEBUF sets the size of the trace buffer in the CONFIG.SYS file.

The parameter 'x' specifies a circular trace buffer size of up to 63K. If you have a TRACEBUF statement without a TRACE statement in CONFIG.SYS, the trace buffer size requested is specified and tracing is turned off (the same as if you specify TRACE=OFF).

If you need to use the System Trace facility, use the largest size, if possible. TRACEBUF=63

TRACEFMT displays formatted trace records in reverse time-stamp order. It is intended to be used to format the trace data so that you can analyse the content of the trace buffer. The most recent entry is displayed first. TRACEFMT numbers each event as it is formatted. The event numbers are unrelated to the trace data, and are useful when discussing a trace with someone else, for easy reference to events.

TRACEFMT works without a filename only if you have a trace buffer defined in the running system.

TRACEFMT works with a filename only if the file is a hex image of a trace buffer from a system for which you have Trace Formatting Files. If the .TFF file is not correct, the entries which are different will be formatted incorrectly with little or no indication of an error.

The file is typically created by the dump formatter by using the command '.TS filename', but TRACEFMT will also save the trace buffer in unformatted form. This is much smaller than the formatted form.

-----

## TRACE and TRACE Processing

The trace command is used to control the system trace.

Command line:	CONFIG.SYS:
TRACE ON	TRACE=ON
TRACE OFF	TRACE=OFF

(you can specify only static TRACE in the CONFIG.SYS file).

The above is optionally followed by one or more major code specifications, or one or more trace definition file specifications, or keywords. Next, you may optionally specify one or more process identifiers. Finally, you may specify that the trace buffer be cleared, and that trace activity be suspended, or resumed.

OS/2 processes TRACE statements in the order in which they appear from any source. TRACE commands in CONFIG.SYS are processed in the order they appear. The effects of the statements are cumulative for the duration of OS/2's execution. If any part of a statement is incorrect, OS/2 ignores the statement.

Process Id is specified by /P:nn,nn,nn (where nn is in HEX!)

Clearing the trace buffer is specified by using /C.

Resuming trace activity is specified by using /R.

Suspending trace activity is specified by using /S.

Major and minor event codes are associated with all trace events. Some of the major codes follow:

Machine Exceptions Major Code: 3

Hardware Interrupts Major Code: 4

Device Helper Routines Major Code: 6

Disk Device Driver Major Code: 7

Major codes may be specified by listing them separated by commas, or as a range, for example, 2-7 specifies codes 2,3,4,5,6,7. Both methods may be combined, as in 5,7,12-18,2,27-32,9 .

If you do not specify TRACE in CONFIG.SYS, event tracing is not started by CONFIG.SYS processing, but may be started later if TRACEBUF has been specified.

Records in the buffer are identified by major and minor codes. Some of the data that may be recorded in the circular buffer includes system events such as interrupts, exceptions, and thread switches.

OS/2 contains a mixture of static tracepoints and dynamic tracepoints.

Static tracepoints are implemented as trace function calls within individual software modules. The TRACE command can be used to turn on and off static tracepoints by specifying them by major code and, optionally, by minor code.

Dynamic tracepoints are implemented by implanting an INT 3 instruction at the specified location, and gathering data when the interrupt occurs. The TRACE command can be used to turn on and off dynamic tracepoints, but only by specifying the module or trace definition file name as a parameter. Dynamic tracepoints cannot be turned on and off by reference to their major codes.

---

## TRACEFMT Processing

When the trace is complete, you can use the trace formatter (TRACEFMT) to format the data into a report. This helps you to isolate causes of problems in the OS/2 system by making the data in the trace buffer available for analysis.

---

## Static & Dynamic Trace, and Files Used

Trace Format Files (.TFF) & Trace Definition Files (.TDF)

Static tracing occurs when a program developer has coded an API call to the system trace interface, which means you cannot specify at what point in the program flow tracing occurs, nor can you control what data is collected.

Trace Format Files are used by TRACEFMT. They specify how the trace data should be formatted. The filename implies which major code is described, and TRACEFMT generates the filename for the .TFF file from the major code of the event about to be formatted. If no description is found, or if the description does not describe all of the trace entry, TRACEFMT defaults to hexadecimal bytes for a default formatting. This will be covered in detail by hands-on exercises.

Trace Definition Files are used for dynamic tracing, and specifying one of them requires you to name the .DLL involved, or KERNEL. You may optionally a type or list of types and a group or list of groups. The .TDF file is used by TRACE to define where to collect data, and what data to collect.

Dynamic tracing occurs when trace definition files ( .TDF ) are used by the TRACE command. The implementation is that OS/2 inserts actual breakpoint instructions at the specified locations, and collects the data specified when the breakpoint is executed. There is no overhead for dynamic tracing when it is not in use, and a technician can be very creative when defining where to collect trace data, and what data to collect. We will create custom dynamic trace entries during hands-on exercises.

The OS/2 static tracepoints do not have associated TDF files, but may have associated TFF files that are used by the TRACEFMT.

---

## Dynamic Trace Processing

Dynamic tracepoints are implemented as Trace Definition File (TDF) entries. The TRACE command can be used to insert (and turn on) a dynamic tracepoint by patching it into its corresponding software module. Dynamic tracepoints are specified by the dynamic link library (DLL) filename and minor code.

Individual dynamic tracepoints can be qualified by separate type and group qualifiers. These qualifiers exist so that you can more easily turn on and off sets of related dynamic tracepoints. For example, all the dynamic tracepoints that are associated with pre-invocation events might have a type of PRE. Similarly, all the dynamic tracepoints that are involved in semaphore processing might have a group of SEM. In the TRACE command syntax, group is considered to have a stronger binding than type. This means that you can ask to turn on all events that are of a specified group that are also of one or more specified types. You do not need to use these qualifiers; they are there simply to make it easier to control related sets of dynamic tracepoints.

TDF files are typically found in the \OS2\SYSTEM\TRACE directory. They are identified by .TDF file name extensions. There are also Trace

Formatting Files (TFF) found within that directory. These files are used by the OS/2 Trace Formatter (TRACEFMT) utility to format the entries that are logged within the system trace buffer.

#### Commonly Used Abbreviations for OS/2 Groups and Types

Groups	Types
FS- file system	API- application programming interface
KBD- keyboard I/O	INT- internal
LDR- resource loader	PRE - pre-processing invocation
LNK- environment management	POST- post-processing invocation
MOU- mouse I/O	
MSG- message management	
MSP- virtual memory management	
NLS- national language support	
PIP- pipe support	
SEL- selector-related	
SEM- semaphore support	
SIG- signal handling	
TIM- timer support	
TK - task management	
TSK- monitor support	
VIO- video I/O	
VM - virtual memory management	

---

## OS/2 Predefined Dynamic Trace Events

The file SYSTEM.TDF file supports dynamic tracing for the following:

```
TRACE ON KERNEL          Major Code: 5 (decimal) 5 (hex)
  Groups: FS, LDR, NLS, PIP, SEL, SEM, SIG, TIM, TK, VM
  Types: PRE, POST, API, INT
  Purpose: Tracepoint definitions for APIs in the OS/2 kernel

TRACE ON DOSCALL1        Major Code: 16 (decimal) 10 (hex)
  Groups: FS, LDR, LNK, MSG, MSP, NLS, SEM, TSK
  Types: PRE, POST, API
  Purpose: Tracepoint definitions for APIs in DOSCALL1.DLL

TRACE ON MONCALLS        Major Code: 16 (decimal) 10 (hex)
  Groups: TSK
  Types: PRE, POST, API
  Purpose: Tracepoint definitions for APIs in MONCALLS.DLL

TRACE ON QUECALLS        Major Code: 22 (decimal) 16 (hex)
  Groups: None
  Types: API, PRE, POST, INT
  Purpose: Tracepoint definitions for APIs in QUECALLS.DLL

TRACE ON SESMGR          Major Code: 23 (decimal) 17 (hex)
  Groups: None
  Types: API, PRE, POST
  Purpose: Tracepoint definitions for APIs in SESMGR.DLL

TRACE ON OS2CHAR         Major Code: 24 (decimal) 18 (hex)
  Groups: KBD, MOU, VIO
  Types: API, PRE, POST
  Purpose: Tracepoint definitions for APIs in OS2CHAR.DLL

TRACE ON PMSHAPI         Major Code: 192 (decimal) C0 (hex)
  Groups: None
  Types: None
  Purpose: Tracepoint definitions for APIs in PMSHAPI.DLL

TRACE ON PMWIN           Major Code: 194 (decimal) C2 (hex)
  Groups: None
  Types: None
  Purpose: Tracepoint definitions for APIs in PMWIN.DLL

TRACE ON PMGRE           Major Code: 195 (decimal) C3 (hex)
```



Groups: None  
Types: None  
Purpose: Tracepoint definitions for APIs in PMGRE.DLL

TRACE ON PMGPI                      Major Code: 197 (decimal) C5 (hex)  
Groups: None  
Types: None  
Purpose: Tracepoint definitions for APIs in PMGPI.DLL

The file SYSTEM.TFF file provides formatting information for OS/2 events.

---

## Steps to Diagnose a Hang

Problems which are called 'hangs' fall into several categories.

The term 'hang' has come into use because there is frequently no way for a user of OS/2 to determine whether the problem is a loop or a wait. The term 'hang' is used in a generic way to mean 'the system does not respond as I expect', or 'I am unable to interact with the system'. The problem may be a loop, or it may be a wait.

Diagnosing any 'hang' will likely be much quicker if the system trace was used to collect appropriate data related to the symptoms.

---

## Steps to Diagnose a Wait

Waits are recognized by the fact that no thread is ready. If the scope of the problem is a single application, we need only find out which thread is expected to run, and then analyse why it will not. If we cannot find out which thread we expect to run, we will need to do the analysis for each thread in the process, which will take somewhat more effort. Frequently, the application can be removed by using the window list to end it. If this has been attempted, and has not worked, one must find out why thread 1 will not execute.

If the scope of the problem is the user interface, one needs to examine it, as discussed above. The workplace shell is discussed elsewhere; remember that from the kernel's viewpoint, it is 'just another application'. This used to be a much more common symptom than in relatively current releases. It was typically noticed on a LAN server, when requesters received normal service responses, but the system administrator could not use the keyboard or mouse.

Frequently, if you haven't a well defined place to start, it works reasonably often to look at the blocking data for all threads, and to choose a resource which is needed by many threads. Pragmatically, if that resource could be made available, many threads would unblock. Therefore, choose one of the more 'popular' block ID's, and proceed to find out what thread owns it, why that thread will not run, and so on. You may need to do this for only one or two resources before you discover the key thread, and can focus your efforts on it.

If the scope of the problem is that OS/2 refuses to run any thread, the problem must be extremely basic, for example, the drive containing SWAPPER.DAT is no longer available due to a hardware problem, or the system has actually terminated, but has been unable to display that fact.

---

## Steps to Diagnose a Loop

Loops are also relatively easy to recognize. When one inspects the collective status of all threads in the system, one thread will be in 'run' status, (if an SMP, one on each processor) and it is likely that many more threads are ready. If the priority of the thread is normal, an application may loop for a long time without the user being aware of the loop, although system performance may suffer somewhat.

To analyse a loop, follow one iteration of it. This is much easier to do with an interactive debugger than it is in a dump.

If the priority of the 'run'ning is in the time-critical class, the dispatcher is designed to prevent OS/2 from dispatching other threads. The looping thread is the cause of the problem, unless the loop is the correct response to another problem. In this case, contact the developer to find out why the thread must be such a high priority, and while you are talking, ask what could cause it to enter a non-ending loop. To diagnose a loop, use an interactive debugger to step through the loop, and try to understand what each conditional jump is really trying to accomplish. You can use an interactive debugger to lower the priority of an offending thread, and 'observe the results'. Recognise that this is quite legitimate, but that the application's integrity may actually depend on the behaviour you have just altered.

---

## Serialization and Priorities in OS/2

This section describes the various ways to serialize access to resources, which is often required in a pre-emptive multitasking environment.

---

### Brute Force Serialization

There are several serialization methods which attempt unilateral control over the dispatcher. Each has its own advantages and disadvantages. They will each be explained here before going on to semaphores, which are much more granular, and therefore less intrusive than these serialization methods.

---

### Uniprocessor Method - Disable Interrupts

There is a way for privileged code to guarantee serialization in a single-processor environment, namely to disable interrupts during the actual inspection and update of the protected resource, and then to enable interrupts promptly. The overhead of this method is practically nil, but it is potentially dangerous because it disables pre-emption, which reduces the responsiveness of the system to the user. It also represents a barrier to running successfully on a multiprocessor, because all other processors are unaffected by this, and it therefore requires the developer to re-examine parts of a program which are no longer serialized, but may well be thought to be properly serialized.

---

### Multiprocessor Methods - Spin Locks

In a multiprocessor environment, there are additional problems, namely how to control the additional processors, which may be executing exactly the same instruction at the same cycle. The solution to successfully serializing access to critical structures is solved by using a special instruction prefix, LOCK, which guarantees that all accesses to memory for the following instruction occur as a unit, with no intervening cycles by other processors, DMA devices, or bus masters.

Instructions such as:

Increment(INC), Decrement(DEC), Add(ADD,ADC), Subtract(SUB,SBB),

Logical operations, (AND,OR,XOR,NOT,NEG),

Exchange(XCHG), Exchange&Add(XADD), Compare&Exchange(CMPXCHG)

can be used to claim a resource, add a node to a linked list, and perform other atomic events which normally require serialization, like selecting a ready thread to run.

**Note:** The 486 and following processors assert the hardware signal 'lock' for XCHG, XADD and CMPXCHG instructions if an operand is in memory, regardless of a prefix.

Each processor will use the appropriate method to attempt its task, and if the condition code does not indicate success, it will simply retry the operation until it does complete. This is called a 'spin loop', and this method of serialization is called a 'spin lock'. It is used when it is expected to be able to access the lock in less time than it will take to save the current context and find another thread to run.

One should not expect to discover the presence of spin locks in a non-multiprocessor environment, because they should always be available, and the spinning should never occur.

---

### DosEnterCriticalSection & DosExitCriticalSection

These API's will serialize all threads in a process. They have no effect on threads in other processes. At the time control returns from DosEnterCriticalSection, no other thread in the process is allowed to execute (there is an exception to this which is when a signal is sent to a process). Looking at the threads' status, one may see 'crt'. This is NOT the thread that entered critical section. Threads in the 'crt' state are ready to run but are temporarily held because of the existence of the thread in critical section. The critical section thread may block, but is the only thread in that process which is allowed to run. Only when that thread issues the DosExitCriticalSection are the other threads released, and again allowed to compete for use of the processor. This is really too much serialization for most situations, because it temporarily disables multithreading in the process, regardless of the other threads' design, or current actual processing.

---

## DosSuspendThread & DosResumeThread

Some applications are designed such that there is a limited number of threads which will access some shared resource, and others never will.

To access a resource in a protected way, one can simply suspend the other threads which represent a possibility of simultaneous update, and leave the remaining threads alone. This is therefore less intrusive than the critical section API's, but still may affect threads which do not represent a threat at this instant, due to other processing, timing, and so on.

DosSuspendThread API will cause a specific thread to no longer compete for the processor, until DosResumeThread is issued. A thread in this situation will have the status of 'frz'. It may not be possible, without an appropriate trace, to find out which thread suspended another.

---

## Semaphores

The least intrusive way to guarantee serial access to a shared resource is to associate a semaphore with it, and to acquire ownership of the semaphore before accessing the resource. The application threads will be suspended only when there is actual contention for the resource.

This does require all of the programmers involved to be careful to request the semaphore before accessing the resource, and to remember to free it when done. The classic solution to this is to build a low-level function which includes the serialization.

Semaphores are of three categories:

1. Kernel Semaphores, or KSEMS.

KSEMS will be discussed later, because we will focus first on items available to the application programmer.

2. 16-bit semaphores.

There are two basic kinds of 16-bit semaphores, and an add-on structure which makes a third type by aggregation.

They are the System Semaphore, the RamSem, and the FastSafe RamSem, which is an accounting structure prefixed onto a RamSem.

3. 32-bit semaphores.

There are two types of 32-bit semaphores, Mutual Exclusion, or Mutex and Event Semaphores. It is also possible to wait on a list of EventSems or MutexSems, but all semaphores in a list must be of the same type.

---

## 16-bit Semaphores

There are three types of 16-bit semaphores, namely system, RAM, and fast-safe RAM semaphores. There are compromises involved in using each.

System Semaphores

These are the most robust of the three, and have the most overhead.

One thread must create the semaphore, with `DosCreateSem`, which has a name in a format similar to a file name, but in root directory 'SEM'. Other threads must open it with `DosOpenSem` to get its handle.

Use is to issue `DosSemRequest`, use the resource, and then to issue `DosSemClear` so that other threads can access the resource. All threads should issue `DosCloseSem` before ending.

If a thread ends while owning a system semaphore, the first requestor is given a return code that indicates the situation, so that it is warned of a possibly incomplete update, and may take whatever action is necessary to recover, or terminate.

To find out which thread owns a system semaphore, display a word at the address provided in the blocking data. The address will be a logical address using a GDT selector, generally 400:xxxx. The 12 low order bits are the slot number of the thread which owns the semaphore. If unowned, the value is zero.

#### RAM Semaphores (RamSems)

At the opposite end of the scale is the extremely fast `RamSem`. Most of the speed comes from the following facts:

- API's use the address of the `RamSem` as the handle.

- OS/2 assumes a `RamSem` is local to a process.

- OS/2 does absolutely no accounting for a `RamSem`.

- OS/2 can not provide any recovery for a `RamSem`.

A `RamSem` is owned by a user thread if the first byte is hex 'FF', otherwise it is not owned by a user thread. Unless you have a trace, there is no way to determine which thread owns a `RamSem`.

#### Fast-Safe RAM Semaphores (FSRamSems)

The `FSRamSem` is a compromise between the two earlier types.

The `FSRamSem` is nothing more than a structure which includes a `RamSem`. The fields of the structure record the process ID (PID) and thread ID (TID) of the thread which owns the semaphore, or zero if unowned. They also include a use count, which is incremented if the owning thread again requests the semaphore. This allows recursive functions to serialize without being blocked, waiting for a resource the thread already owns.

The `DosFSRamSemRequest` API is used to request the semaphore. It returns when the resource is owned by the thread.

The `DosFSRamSemClear` API is used to release the semaphore. If the use count is not zero after being decremented, the semaphore is NOT released. There must be as many 'Clear' as 'Request' API calls to actually release the semaphore, and allow other threads to compete for it.

---

## 32-bit Semaphores

There are two classes of 32-bit semaphores, private and shared. There are three types of semaphore in each class, Event, MUTual EXclusion, and multiple wait semaphores.

MUTEX semaphores correspond to one of the most common uses of the 16-bit semaphores, namely to allow competing threads to mutually exclude others from accessing a shared resource.

A MUTEX semaphore includes the slot number of its owner, if owned, or zero if unowned.

An EVENT semaphore contains a 'post count' which is incremented each time it is POSTED, and decremented each time a WAIT for it is completed successfully. This type provides a way to insure that some processing occurs exactly once for each POST.

A multiple wait semaphore is nothing more than a list of semaphores, of the same type. A thread may wait on either 'ANY' or 'ALL' of the semaphores in the list.

All Semaphores must first be created with `DosCreate???Sem`, where '???' is the semaphore type. Other processes must open them with `DosOpen???Sem` to have access to them. Private semaphores have a null pointer to their name, and thus no name. Public ones have a name in the same format as that used for the 16-bit semaphores. `DosClose???Sem` is used when a thread is through using it.

DosRequestMutexSem and DosReleaseMutexSem are used to access the mutual exclusion semaphores.

DosPostEventSem and DosWaitEventSem are used to access an event semaphore. DosResetEventSem will allow immediate access, and return the post count, which is cleared by this API.

DosQuery???Sem will allow the retrieval of information about each type of semaphore.

DosAddMuxWaitSem and DosDeleteMuxWaitSem are used to add and delete semaphores from a multiple wait semaphore list, respectively.

-----

## Dispatching Priorities

This section describes how the priority of a thread is set, and defines what the classes mean for debugging.

-----

## The Dispatcher, Priorities, and Dispatching Classes

The dispatcher's task is to give control to the proper thread. The definition of 'proper' thread can be difficult to state. My approach to this problem is to state the obvious cases, and then to focus on what is left. In a sense, this discussion parallels the logic in the dispatcher.

1. Idle Class.

No other class will be pre-empted in order to run an idle class thread. The notion of starved, and the MAXWAIT parameter do NOT apply to Idle Class threads. OS/2 by design will not execute a ready Idle Class thread as long as threads in other classes are ready.

2. Regular Class.

Most threads are expected to be in this class. All dispatching options and parameters apply to scheduling this thread.

3. Time-Critical Class.

As long as any thread in this class is ready, OS/2 will give control to it. By design, this may prevent threads in other classes from running. You cannot use priority as a serialization method.

For example, a page fault will result in temporarily blocking this priority thread.

4. Fixed-High, or Server Class.

The threads in this class are at a somewhat higher priority than those in the regular class which do not have the focus, but below time-critical.

**Note:** The numbers above are what the programmer specifies in DosSetPriority, or the 16-bit API DosSetPrty, and are what is returned by DosGetPriority. OS/2 processes these class numbers to create an internal dispatching priority. There are 32 priority levels in each class, which range from 00 to 1F. The priority levels, or deltas, stay the same as the programmer specified initially, if PRIORITY=ABSOLUTE is specified.

The internal priorities have a range from 01 to 08, with 01 usually used for idle-class threads, and 08 usually used for time-critical threads. If PRIORITY=DYNAMIC was specified or defaulted, there are priority boosts given for the following reasons:

Being the foreground process; and for owning the keyboard;

Yielding the processor before the end of the time slice

IO completion

Being 'starved', that is, ready status and not dispatched for MAXWAIT seconds.

Dispatching is the process of finding the correct ready thread, and then giving control to it. Within each class, the priority delta is used to choose which thread should have control. When several ready threads have the same priority, control is given in turn to each of them, based on the TIMESLICE parameter. The minimum value of this parameter is the duration of the priority boosts which may be applied. The maximum value is the longest a thread can execute before being pre-empted for other threads which have the same internal dispatching priority.

As long as a group of threads at some priority use all the processor, control is not given to lower priority threads. What happens is that the other waiting threads become 'starved' after MAXWAIT seconds have elapsed, and their priority increases until they receive at least a minimum timeslice.

Idle-class threads are never starved, and so will not receive this boost.

**Note:**

When running in the kernel and device drivers, pre-emption can not occur. Threads must explicitly give up their time-slice to give other threads an opportunity to run.

## How to Display Dispatching Priority

Use the '.p' command on the slot of interest, and find the pTCB, which is the linear address of the Thread Control Block.

For slot F, below, we see the following FOR .p output:

```

*
Slot Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
000c 0005 0000 0005 0001 blk 0200 7cf7f000 7d1858a4 7d16a0d8 1eb8 00 pgma
0008 0005 0000 0005 0002 blk 081f 7cf77000 7d1858a4 7d169a28 1ea8 00 pgma
000e 0005 0000 0005 0003 blk 021f 7cf83000 7d1858a4 7d16a430 1ea8 00 pgma
000f 0005 0000 0005 0004 blk 061f 7cf85000 7d1858a4 7d16a5dc 1ea8 00 pgma
0010 0005 0000 0005 0005 blk 0200 7cf87000 7d1858a4 7d16a788      00 pgma
000d 0006 0000 0006 0001 blk 0200 7cf81000 7d1860d0 7d16a284 1eb8 00 pgmb
000a 0006 0000 0006 0002 blk 021f 7cf7b000 7d1860d0 7d169d80 1eb8 00 pgmb
0013 0006 0000 0006 0003 blk 0800 7cf8d000 7d1860d0 7d16ac8c 1eb8 00 pgmb

DB  %7D16A5DC+164  L 6              would show ( for slot f )
                                   SEE CAUTION, BELOW
%7D16A740  02  1F  xx  xx  1F  06
  class /    /  -----  the word value 061F
  level /    /  actually used by the dispatcher
```

CAUTION: the offset used is correct for WARP CONNECT, but the addresses are what were used in OS/2 2.11, so this is a somewhat mixed example. Any offset in any control block may change any time a fix or new version is installed. Please refer to the [Thread Control Block](#) in the System Diagnostic Reference for offsets relating to other versions.

The first byte is the programmer's priority class, ranging from 1 to 4. The second byte is the level within the class, ranging from 00 to 1F. The third and fourth bytes are not useful. The fifth and sixth bytes are OS/2's computed dispatching priority. This field is a word, so the high order part is byte 6.

081F is the highest possible value.

0100 is the lowest possible value.

On a uniprocessor, using DosSetPriority to make yourself time-critical at the highest delta value would give you an extremely good chance of not being pre-empted, and was occasionally misused for serialization. This will never work on a multiprocessor, and is risky even on a uniprocessor, because a page fault will cause you to lose control while the page is processed, just as doing I/O to a file will cause a thread to block if access to the actual device is required.

## The Status of a Thread

A thread can be in one of several states. The following list is an attempt to list all the possible states, and to briefly discuss each.

run	This thread is currently executing.
rdy	This thread would like to run, but higher priority thread(s) are executing, which prevents this thread from running.
blk	This thread is blocked. Use the '.pb' command to find out what resource (block id) it is blocked on (waiting for).
crt	This thread cannot be run because another thread in this process is currently in a critical section.
frz	This thread is frozen, that is, some other thread has called the API DosSuspendThread and passed the ID of this

thread. This thread cannot execute until some thread issues DosResumeThread to inform OS/2 that this thread is once again eligible to be dispatched. There is no way to discover what thread suspended it.

-----

## A Form to Use For Unwinding Stacks

Frame at	Caller's Frame	Return Offset	Selector	Parameters			
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____	_____	_____

-----

## Advanced Guide to Hang Analysis

What is a *hang*?

It's an external symptom or a user perception that little or no work is being done. It could be a case of extremely poor performance. The hang symptom categorises itself into three distinct cases:

#### Wait

Threads and processes are not being dispatched by the Operating System. Thread status gives the initial clue. Use of the **.P** command determines status.

##### Blocking

Threads may wait voluntarily for a resource or an event, in which case they will probably be **Blocked**. They might poll for the resource, in which cases they will cycle through blocking, being ready and running until the resource becomes available.

A notoriously problematic case of Blocking is the **deadlock**. This is where two threads are each own exclusively a resource and block waiting for ownership of the other's resource.

##### Suspension

Another thread may have deliberately debarred a thread from being scheduled, in which case we will see the waiting thread in a **cr**t or **fr**z state.

##### Pre-emption

Another thread monopolises the system. Typically the hanging threads will be ready for dispatch (**rdy**), but will never or rarely receive a minimum time-slice. We look for running and ready threads with an excessively high priority. The **.P** will give the calculated priority of each thread.

##### Disabled wait

Looks rather like a H/W Freeze. Last instruction executed was a **HLT** having sometime previously disabled interrupts using **CLI**. This usually happens only when ring 0 code detects a terminal condition. One would hope that some form of diagnostic information had been displayed prior to this particularly if NMIs have been disabled also! If NMIs have not been disabled then an artificially generated channel check may be used to cause an NMI, which would allow one to break into the kernel debugger. However the KDB to allows only one NMI channel check per boot. If NMIs are disabled then H/W analysis techniques may be the only recourse.

#### Loop

A thread is running more or less permanently. It's state will mostly be **rdy** or **run**. Similar analysis techniques to trap apply here. We examine the registers of the running thread by using **.R**. From there we can determine in which module the thread is running by looking at the owner of the executing code segment. If necessary we unwind the stack and determine the caller etc.. Analysis is very similar to trap analysis.

#### H/W Freeze.

The processor fetch-execute cycle has been suspended. Not even an NMI interrupt will resume instruction fetch. This is almost certainly a hardware problem. Timing/clocking problems caused by incompatible cards, overloaded busses, incorrect bus terminations, faulty processor or controller chips. Use H/W techniques such as an ICE machine or Logic Analyser.

The following Theory Topics are now covered in detail:

- [The Wait Condition - Diagnostic Techniques](#)

This is further subdivided into two topics of discussion:

[Memory Management and Ownership Topics](#)

[Thread Scheduling and Dispatching](#)

- [Program Design Issues and Weaknesses](#)

The final section of this Guide is a collection of [Worked Examples](#) that explore memory management, the File System, Presentation Manager and Ring 0 Loops From a Dump.

---

## The Wait Condition - Diagnostic Techniques



In most problem analyses the question of memory ownership or use will arise. For example:

*To which module does this instruction belong?*

*Which process executed this module?*

*Who allocated this storage?*

*Who passed these parameters?*

*What control block does this address point to?*

In fact the frequency with which this question is asked makes it a fundamental aspect of analytical technique.

For hang analysis this is no less true:

In the case of loops, analysis proceeds in a similar manner to that of traps.

In the case of waits, a key piece of information is the [BlockId](#). In many cases this is an address of a system control block that relates closely to the reason for waiting. Discovery of the owner of storage pointed to by a BlockId is therefore of prime interest.

We start by reviewing memory management in OS/2 and in particular memory ownership.

-----

## Memory Management and Ownership Topics

Memory allocation in a demand-paging virtual storage operating system such as OS/2 embodies the allocation of a number of system resources with certain attributes applied:

### Resources

Virtual address space

Real address space

Auxiliary address space (SWAPPER)

### Attributes

Exclusion (privacy)

Inclusion (sharing)

Owner (Where it was allocated from or who it was allocated to)

Requestor (who made the request on behalf of the owner)

Permissions (Read-only, Read/Write, Executable)

Use of the resources and the attributes applied is tracked by the system in its VM control blocks. The most important of these are:

VMAH	The Virtual Memory Arena Header Record
VMOB	The Virtual Memory Object Record
VMAR	The Virtual Memory Arena Record
VMCO	The Virtual Memory Context Record
PF	The Page Frame Structure
VP	The Virtual Page Structure

-----

## Virtual Address Space Arenas and Regions

OS/2 partitions the 4G virtual address space into three types of arena:

System

Shared

Private

The system arena is common to all processes. It starts at the 512M boundary and occupies the address space up to 4G. Only system code (and device drivers) can access data in the system arena directly. User code must use APIs invoked by the call gate mechanism to access system arena code and data. Nearly all system arena data is global: that is, managed by a common set of page tables, whatever the current thread/process context. The exception to this is in the memory area mapped by selector 30. Page table entries are adjusted as part of context switching so that selector 30 addresses the current PTDA, TCB and TSD.

The shared arena address range is common to all processes, but it comprises data that is both global and instance. Instance data occurs where a separate set of page table entries are used per context to map the same linear address range.

Instance data is used when the same type of data needs to be allocated as multiple private copies to each process. An example of this would be a logical screen buffer. The shared arena starts initially at the 304M boundary and ends at 512M. User programs may access the shared arena. DLL code and data is located in the shared arena. DLL code segments are always global, but DLL data segments may be instance or global and are usually a mixture of both.

The shared arena is further subdivided into a number of regions:

<i>Region</i>	<i>Description</i>
Protected	<p>This region is reserved for protected data segments of protected DLLs. In General 16- and 32-bit applications do not have addressability above the 448Mb boundary. Potentially 32-bit applications are able to modify all read/write global data, whether intended by the owning DLL or not. The Protected region is provided so that Protected DLLs can isolate their data from general access. Only Protected DLLs have addressability to the protected region by being assigned data selector 63.</p> <p>32-bit DLLs become protected through use of the protect option at compile time.</p> <p>16-bit DLLs may also use the protected region, if explicitly coded to do so and listed in CONFIG.SYS using:</p> <pre>PROTECT16=dll1,dll2,...</pre> <p>The Protected Region may be subsumed into the Based Region (see below) by coding in CONFIG.SYS the <b>NOPROTECT</b> option on the <b>MEMMAN</b> statement.</p> <p>The default is <b>MEMMAN=PROTECT</b></p> <p><b>Note:</b></p> <p>From OS/2 Warp V3.0 fix pack 19 and OS/2 Warp V4.0, the Protected Region has been absorbed into the Based Region. The system behaves as if <b>MEMMAN=NOPROTECT</b> is in effect and <b>MEMMAN=PROTECT</b> has no effect.</p>
Based	<p>The Based Region is reserved for non-protected DLLs that have a preferential base address assigned by the linkage editor by using the <b>BASE</b> option.</p> <p>The purpose of the Based Region is to improve performance of module loading, by avoiding the need for the System Loader to do fix-up processing.</p> <p><b>Note:</b></p> <p>Under OS/2 2.x, <b>MEMMAN=NOPROTECT</b> would cause the Based and Packed Regions to move up 64M bytes - effectively giving another 64M bytes for general purpose use in the Shared Arena.</p>
Packed	<p>The Packed Region is reserved for 16-bit DLL code segments. Within the Packed Region the <a href="#">Compatibility Region Mapping Algorithm</a> does not apply. Code segments are packed contiguously to make best use of physical pages.</p> <p>Potentially, tiny DLL code segments can deplete physical storage very rapidly if not packed. However, when packing is used there is no general algorithm that will convert 16-bit addresses into 32-bit addresses within the Packed Region. The system has to scan</p>

the **LDT**, over the Packed Region, to make this conversion.

Packing may be disabled by specifying the **NOPACK** option of the **MEMMAN** statement in CONFIG.SYS. The default is **PACK**. When packing is disabled up to 32M bytes is made available to the Global Shared Region.

**Notes:**

Under OS/2 2.x only **MEMMAN=NOPACK** would tend to reduce the Swapper Size where a great many 16-bit code segments are in use. This is because code segments outside the Packed Region are normally discardable (they are not swapped). Within the packed region they are swappable since a 4K page may contain code from a number of different modules.

Under OS/2 2.x **MEMMAN=NOPACK** would provide up to 32M bytes extra virtual address space for general purpose use in the Shared Arena.

Packing does not affect the availability of LDT selectors for allocations in the Packed Region, just the base linear address boundaries on which they are deployed.

Packing should not be confused with either the LINK386 PACK option or the PACK.EXE utility.

From OS/2 Warp V3.0 fix pack 19 and OS/2 Warp V4.0 the packed region has been reduced to 16Mb.

Global Shared

This region has a lower boundary at 320M bytes and includes the Packed, Based and Protected Regions. This is reserved for Global Read-Only allocations only. Since no Read/Write data is allocated in the Global Shared Region some page table economies are possible. Also process context switching performance is improved.

**Notes:**

The Global Shared Region is not configurable.

The Global Shared Region is only implemented in OS/2 WARP version 3.

Under OS/2 2.x Read/Write segments would be allocated in the Based Region.

Read/Write Basing

The Read/Write Basing region is the preferred region for locating Read/Write DLL data segments where a base address has been assigned to a DLL module by the linkage editor. The purpose of this region is to keep Read/Write segments out of the Global Shared Region and thus retain its performance advantages. It also places an upper bound on the location of dynamic shared allocations, namely the Minimum Read/Write Basing Region address.

**Notes:**

The Read/Write Basing Region is not defined in OS/2 versions prior to version 3.

Based DLLs under OS/2 2.x, by preference, have their segments loaded sequentially starting with segment 1 at the base address. With the implementation of the Global Shared Region only Read-Only segments can be loaded sequentially from the base address.

Expansion

The Shared Arena is an *expand-down* arena, that is, allocation searches for free regions from the high addresses to low. The Shared Arena therefore expands from the minimum Read/Write Basing Region address towards the highest upper bound of all the Private Arenas. This area is the expansion region for both the Shared Arena and all the Private Arenas.

The Shared Arena will not contract to an address higher than the minimum Read/Write Basing address.

**Note:**

The expansion region for OS/2 2.x is from the lower boundary of the Packed Region, if present. If not, then from the lower boundary of the Protected Region, if present. If both the Protected and Packed Regions are removed (using **MEMMAN=NOPACK,NOPROTECT**) then expansion occurs from the top of the Shared Arena.

Each private arena occupies the lowest range of virtual address space from 0 - 64M bytes expanding up to a maximum of 304M bytes, the minimum Read/Write Basing address. None of the Private Arenas will be allowed to expand beyond the lowest allocation in the Shared Arena, that is Private and Shared Arenas may not overlap.

In general each process uses a separate set of page table entries to map each page of its private arena. Thus the data in the private arena is private to each process. Code (.EXE files) however is treated differently. Since code is read only an economy is made whenever more than one process runs the same .EXE. Where this happens the same page table entries are used among the processes sharing the common .EXE file. User programs may only access the private arena of the process they are running in (a special exception to this is possible through the DosDebug API by defining memory aliases).

Virtual Memory Arenas and Regions may be presented pictorially as in the following diagram.

**Note:**

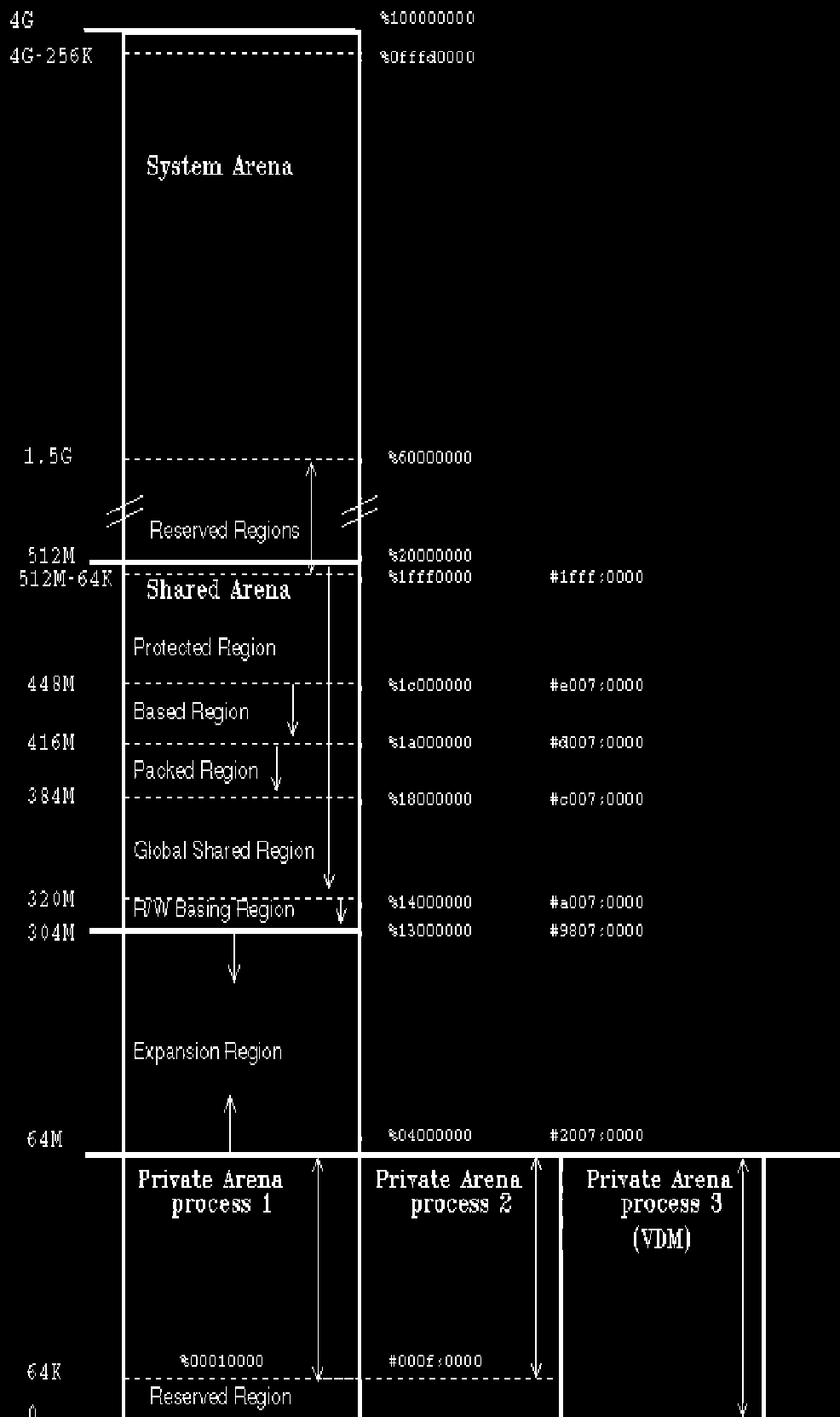
Some regions of the 4G address space are reserved. This is done for a variety of reasons which include:

- H/W and BIOS restrictions

- Enforced segregation between Arenas

- Guaranteed reserved address ranges.

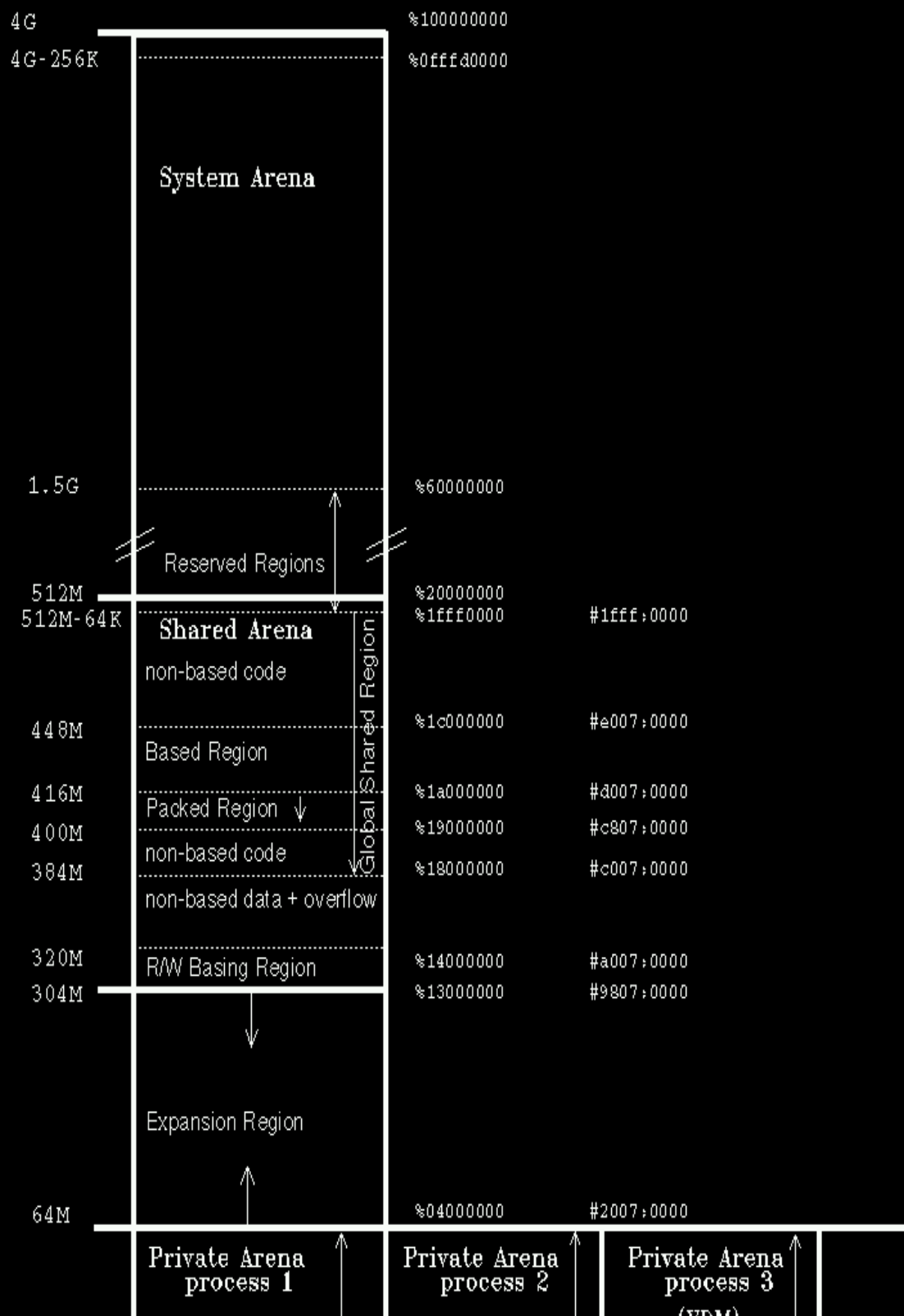
# Virtual Address Space Regions





# Virtual Address Space Regions

(OS/2 4.0 and 3.0 Fix Pack 19)



---

# Virtual Address Space Management

Each of the three types of arena discussed in the previous section is managed by:

An Arena Header Record (**VMAH**)

A Sentinel Arena Record (**VMAR**)

The **VMAHs** are maintained in a double-linked list. They contain information about the extent to which an arena has been used. Of particular interest are the following fields:

+0x0	Pointer to the next VMAH
+0x4	Pointer to the previous VMAH
+0x8	Pointer to the Sentinel Arena Record for this arena
+0xc	Pointer to the VMAR adjacent to the 1st free area.  In the case of expand down arenas (the shared arena), this is the VMAR for the region of memory allocated above the first free area below the Minimum Read/Write Basing region.  In the case of expand-up arenas (system and private) this is the VMAR for the region of memory allocated just below the lowest free area.
+0x20	Current minimum linear address allocated.
+0x24	Current Maximum linear address allocated.

**VMAHs** are located:

- at \_ahvmSys for the Shared Arena
- at \_ahvmhShr for the High Memory System Arena
- at \_ahvmShr for the System Arena
- imbedded at +0x40 in each PTDA for Private Arenas

Arena Records (**VMARs**) are used to describe virtual storage reservations. These are described in more detail in [Virtual Memory Arena Records](#), below.

A special form of the **VMAR** is the Sentinel Arena Record. This serves two purposes:

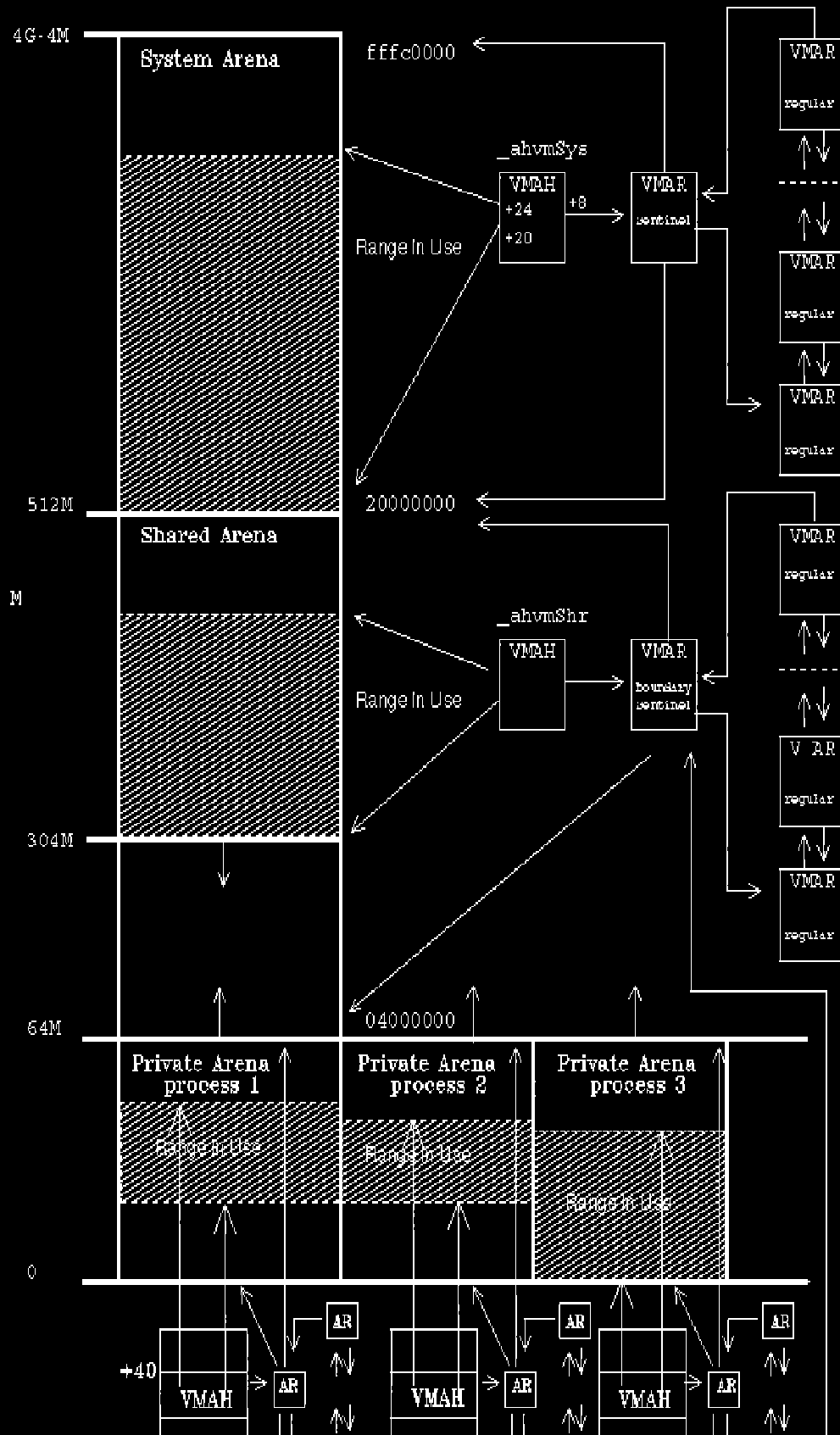
- To track the theoretical size limits of an Arena
- To act as the head to a double-linked list of Regular **VMARs**, each of which describes a specific allocation.

The sentinel **VMAR** for the Shared Arena is called the Boundary Sentinel, since it determines where the (dynamic) boundary between shared and private arenas lies. The boundary is adjusted to reflect the current highest private arena address.

The manner in which **VMARs** and **VMAHs** are organised to manage the three types of arena is shown in the following diagram:



# Virtual Address Space Management



Arena records appear in a number of guises depending on the area storage they describe and whether the storage is shared, instance or private data. They are formatted by the KDB and DF **.MA** command. **.MA** takes either the handle or address or the VMAR as a parameter, or if no parameters are specified then the entire VMAR table is formatted.

The fields of principle interest are:

cpg The number of pages (4K bytes) allocated or reserved.

va The address of the first page in the reservation.

hob The handle of the **VMOB** that occupies the virtual address range covered by va and cpg.

The right-hand column gives descriptive information about the use of the address range in a VMAR. Of particular interest are:

`sel=ssss` Indicates system storage mapped by a GDT selector.

hco=hhhh

Indicated shared global storage. The **hco** is the handle of the **VMCO** at the head of the list representing accessors to an allocation in the Shared Arena.

hptda=pppp

Indicated private memory allocated in the private arena of a process whose PTDA handle is pppp.

VMOBs are 16-byte records allocated contiguously in a table in system memory. Each table entry is numbered from 1 and is referred to as a memory object handle, or more simple as a **hob**.

VMOBs are used to store information about the allocation request. Of particular interest are:

- The Requestor
- The Owner
- The Permissions

The VMOB also has links to other related control blocks. Of these the important ones are:

The associated VMAR,  
the associated VMCOs,  
and associated VMOBs.

VMOB is formatted by using the KDB or DF **.MO** command. **.MO** takes either the handle or address or the VMOB as a parameter, or if no parameters are specified then the entire VMOB table is formatted.

```
##.mo
hob har hobnxt flgs own hmte sown,cnt lt st xf
0001 0001 fec8 0000 fff1 0000 0000 00 00 00 00 vmob
0002 0002 fec8 0000 ffe3 0000 0000 00 00 00 00 vmar
0003 0003 fec8 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
0004 %fff13238 8000 ffe1 0000 0000 00 00 00 00 vmah
0005 %fff13190 8000 ffe1 0000 0000 00 00 00 00 vmah
0006 %fff0a891 8000 ffa6 0000 0000 00 00 00 00 mte doscalls.dll
0007 0006 0000 0000 ff6d 0000 0000 00 00 00 00 doshlp
0008 0007 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
0009 0008 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000a 0009 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000b 000a 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000c 000b 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000d 000c 0000 0325 ffba 0000 0000 00 00 00 00 lock
000e 000d 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000f 000e 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
0010 008f 0000 402c 00ae 0115 0000 00 00 00 00 priv 0002 c:\pmsHELL.exe
0011 0010 0000 0000 ff37 0000 0000 00 00 00 00 romdata
0012 0011 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
```

One VMOB is formatted per line with the hob in the left-hand column.

The owner is shown under the **own** column and is given as a hob that is associated with and uniquely identifies where the allocation is made from. For example:

Memory dynamically allocated within a Private arena uses the handle of the PTDA (**hptda**) as the owner.

The PTDA has a number of characteristics that make it an ideal choice for an owner:

Each process has a unique PTDA

The PTDA is the central control block from which all information about a process is obtained.

Each PTDA is allocated from a unique memory object so has a unique hob, which is defined to be the hptda.

For storage allocated for a load module segment the module MTE handle (**hmte**) is used.

The MTE has a number of characteristics that make it an ideal choice for an owner:

Each loaded module is represented in the system by an unique MTE.

Each MTE is allocated from a unique memory object so has a unique associated hob, which is defined to be the hmte.

Memory allocated in the shared arena which is not specific to a particular process uses the following conventions for owner:

- For DLL instance and global data the owner is the hmte of the owning DLL.
- For Giveable shared storage, the owner is (0xfff5).
- For Gettable shared storage, the owner is (0xfff6).
- For Giveable and Gettable shared storage, the owner is (0xfff7).

- For Named Shared Storage, the owner is (0xff82).

See DosAllocSeg, DosGiveSeg, DosGetSeg and DosAllocSharedMem APIs in the Control Program Programming References for OS/2 1.x, 2.x and 3.x.

Memory allocated or suballocated from the system arena uses an artificial system owner id (ffxx) that doesn't actually correspond to a VMOB but is a conventional handle used to indicate the type of system object which has been allocated. An example this is hob 1 which is the table of VMOBs.

The requestor's id is shown in the **hmte** column. This field is either:

The hmte of the module making the request.

An associated system object

zero where there is no associated requestor.

To the right of each line appears a textual interpretation of the **own** and **hmte** fields.

The handle of the associated **VMAR** is shown in the **har** column.

Associated VMOB's that share the same virtual address (that is, instance data) are linked from the **hobnxt** field.

Not every VMOB is linked to an associated VMAR, as seen in hobs 4 and 5 in the example. These are known as pseudo-objects. They are used for some small system control blocks that are allocated, as required, from system storage but are too small to warrant the overhead of the minimum allocation of 1 page, which an arena records implies. PTDA's and MTE's are the most frequently encountered pseudo-objects. The **va** field replaces the **har** and **hobnxt** and points directly at the object itself.

## The Virtual Memory Context Record

VMCOs are small control blocks that serve as extensions to the VMAR for shared arena, shared data. Whenever a process is given access to a shared global data object (most frequently DLL code and global data) then a VMCO is used to record the handle of the process (hptda) of the accessing process. Each process that shares a global data object will have a VMCO chained in a single-linked list from the object's VMAR.

VMCOs may be formatted using the KDB and DF **.MC** command, however they are usually displayed with their corresponding VMOB and VMAR by using either the **.MOC** or **.MAC** commands.

```
#.mac 297
```

```
*har    par      cpg      va      flg next prev link hash hob    hal
0297 %feef2904 00000660 %11fb0000 369 0312 0295 0000 009e 02a1 0000 hco=0057f
hob    har hobnxt flgs own hmte sown,cnt lt st xf
02a1 0297 0000 4a2d fff5 0302 0000 00 00 00 00 give
hco=057f pco=ffe70b96 hconext=00241 hptda=06d1 f=1e pid=0059
hco=0241 pco=ffe6fb60 hconext=004ce hptda=04b2 f=1e pid=0043 c:pmspool.exe
hco=04ce pco=ffe70821 hconext=0034c hptda=05c3 f=1e pid=0016 c:pawn.exe
hco=034c pco=ffe70097 hconext=001e4 hptda=04ca f=1e pid=000f c:pmsshell.exe
hco=04ca pco=ffe7080d hconext=00348 hptda=05c3 f=16 pid=0016 c:pawn.exe
hco=0348 pco=ffe70083 hconext=0017a hptda=04ca f=16 pid=000f c:pmsshell.exe
hco=017a pco=ffe6f77d hconext=00177 hptda=03d9 f=16 pid=000b c:spdaemon.exe
hco=0177 pco=ffe6f76e hconext=00148 hptda=03ec f=16 pid=000c
hco=0148 pco=ffe6f683 hconext=000b2 hptda=03c9 f=16 pid=000a c:ddaemon.exe
hco=00b2 pco=ffe6f395 hconext=00083 hptda=0359 f=16 pid=0009 c:harderr.exe
hco=0083 pco=ffe6f2aa hconext=00081 hptda=02e1 f=16 pid=0007 c:landll.exe
hco=0081 pco=ffe6f2a0 hconext=0007d hptda=02ad f=16 pid=0006 c:lanmsgex.exe
hco=007d pco=ffe6f28c hconext=00037 hptda=027a f=16 pid=0008 c:pmsshell.exe
hco=0037 pco=ffe6f12e hconext=00000 hptda=02ac f=16 pid=0004 c:gambit.exe
```

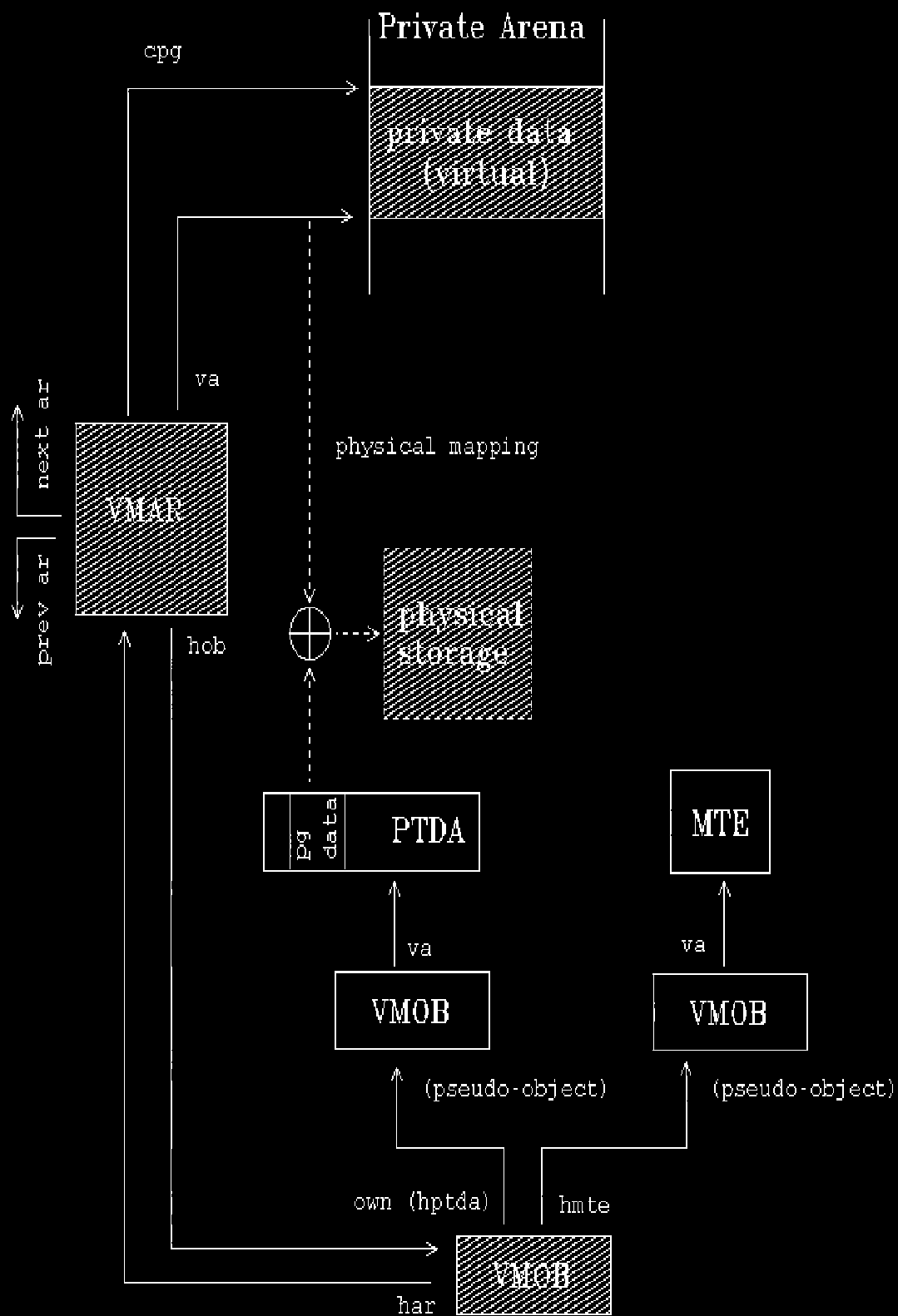
## Private Arena Private Data

Private data, that is data in a Private Arena not accessible from any other context, is managed by VMARs and VMOBs as depicted by the

following diagram.

Control blocks and data that directly represent the allocation are shown shaded.

# Private Arena Private Data



---

## Private Arena Shared Data

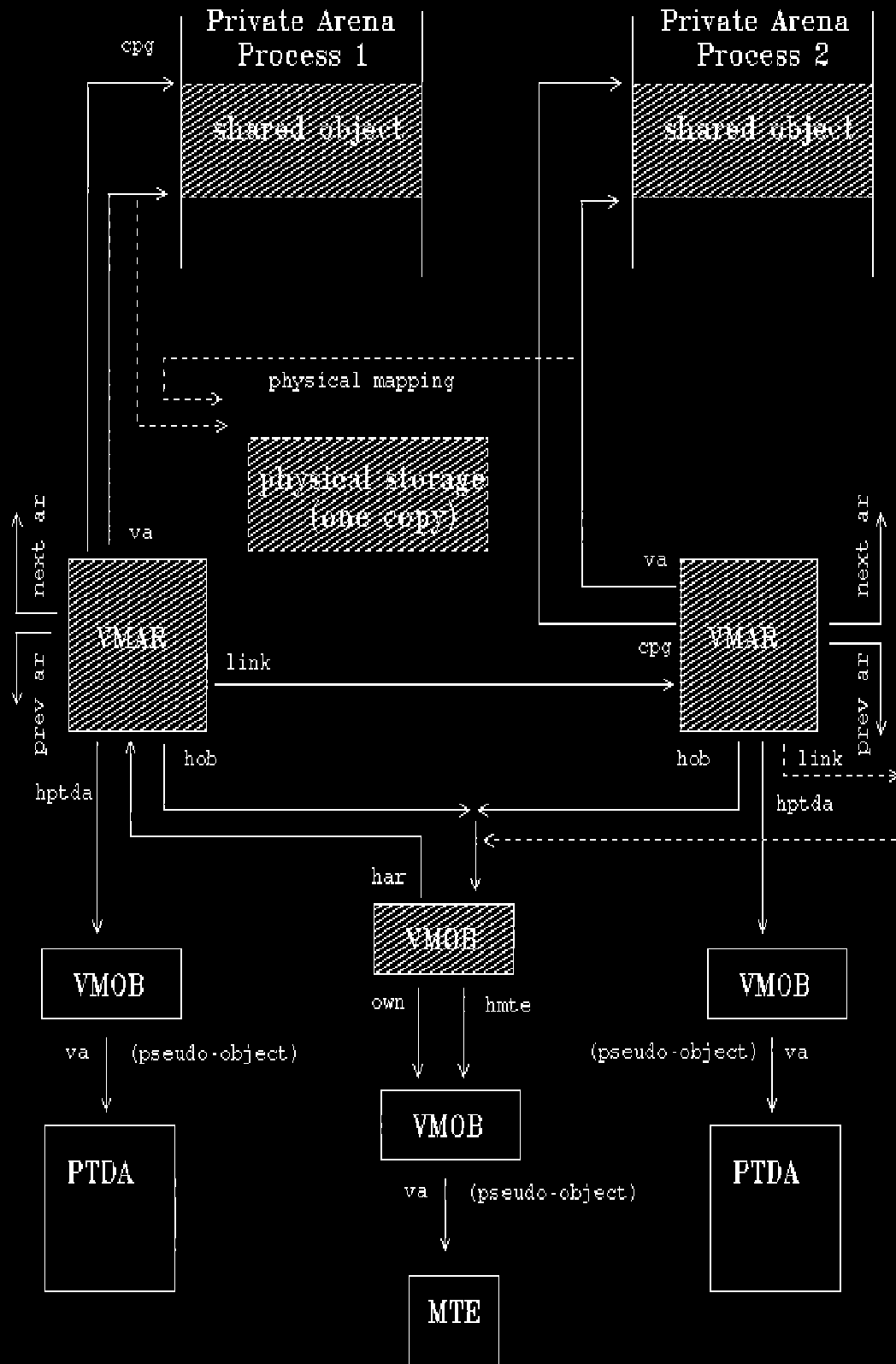
This is the case where .EXE program Read/Only segments are shared across multiple Private Arenas.

The following diagram depicts this situation.

Note that only one VMOB is used, but there are multiple VMARs, one for each process accessing the allocation. Each VMAR is linked using the link field.

Control blocks and data that directly represent the allocation are shown shaded.

## Private Arena Shared Data





-----

# Shared Arena Global Data

DLL Global Data and Named Shared, Giveable and Gettable allocations are potentially shareable among multiple processes. With these types of allocations data and address range is common to all who access it. Those that are given access are recorded by the VMCO chain.

With this type of allocation, there is only one VMAR, and VMOB.

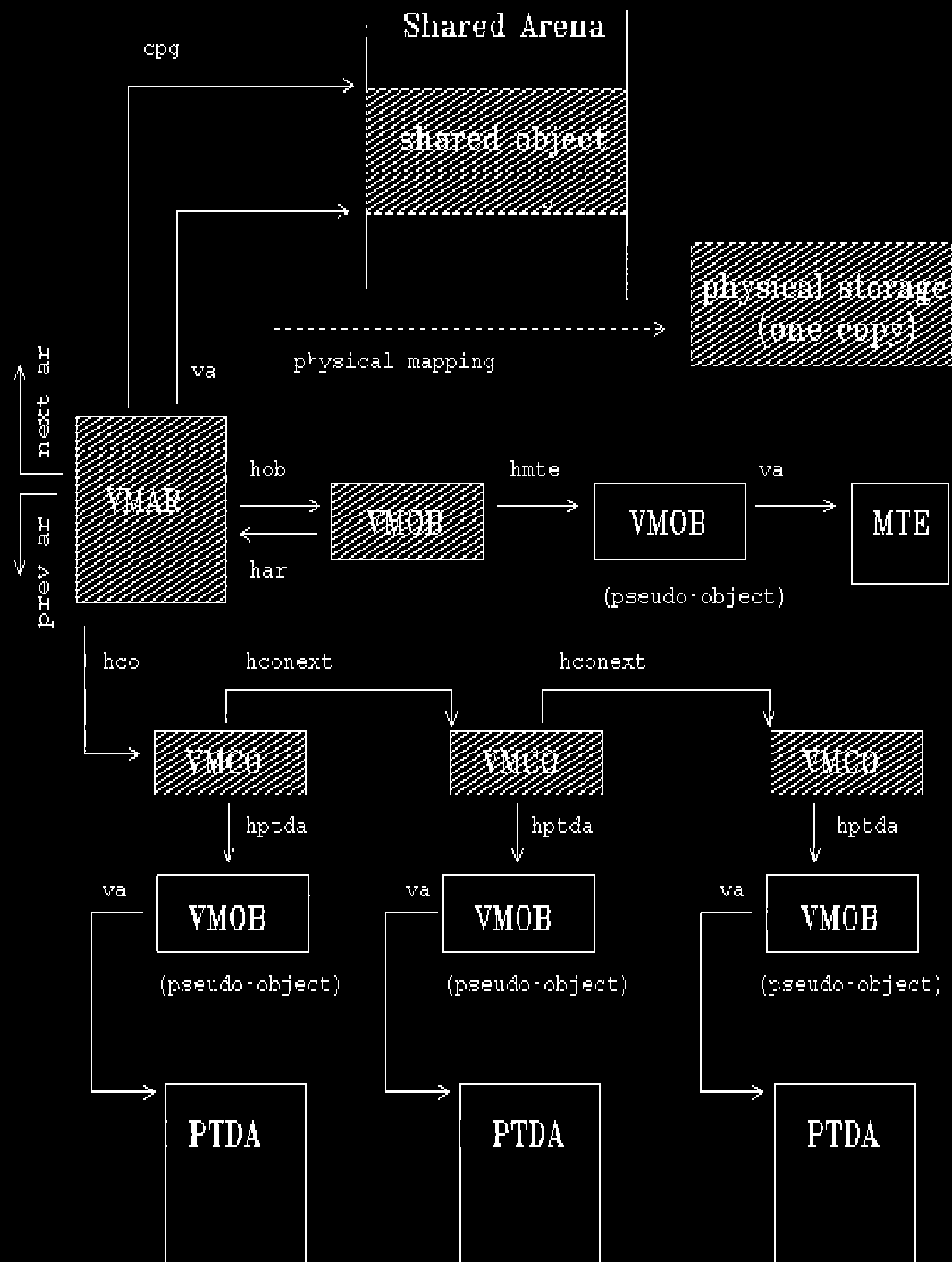
Note that the own field of the VMOC, which is interpreted on the right hand side of the .MO display, may be one of five possibilities:

hmte	Data is Global Data or Code segment of a DLL
Give	Data is allocated with the Give attribute
Get	Data is allocated with the Get attribute
GiveGet	Data is allocated with both the Give and Get attributes.
Mshare	Data is named shared storage.

The following diagram depicts this situation.

Control blocks and data that directly represent the allocation are shown shaded.

# Shared Global Data



-----

## Shared Arena Instance Data

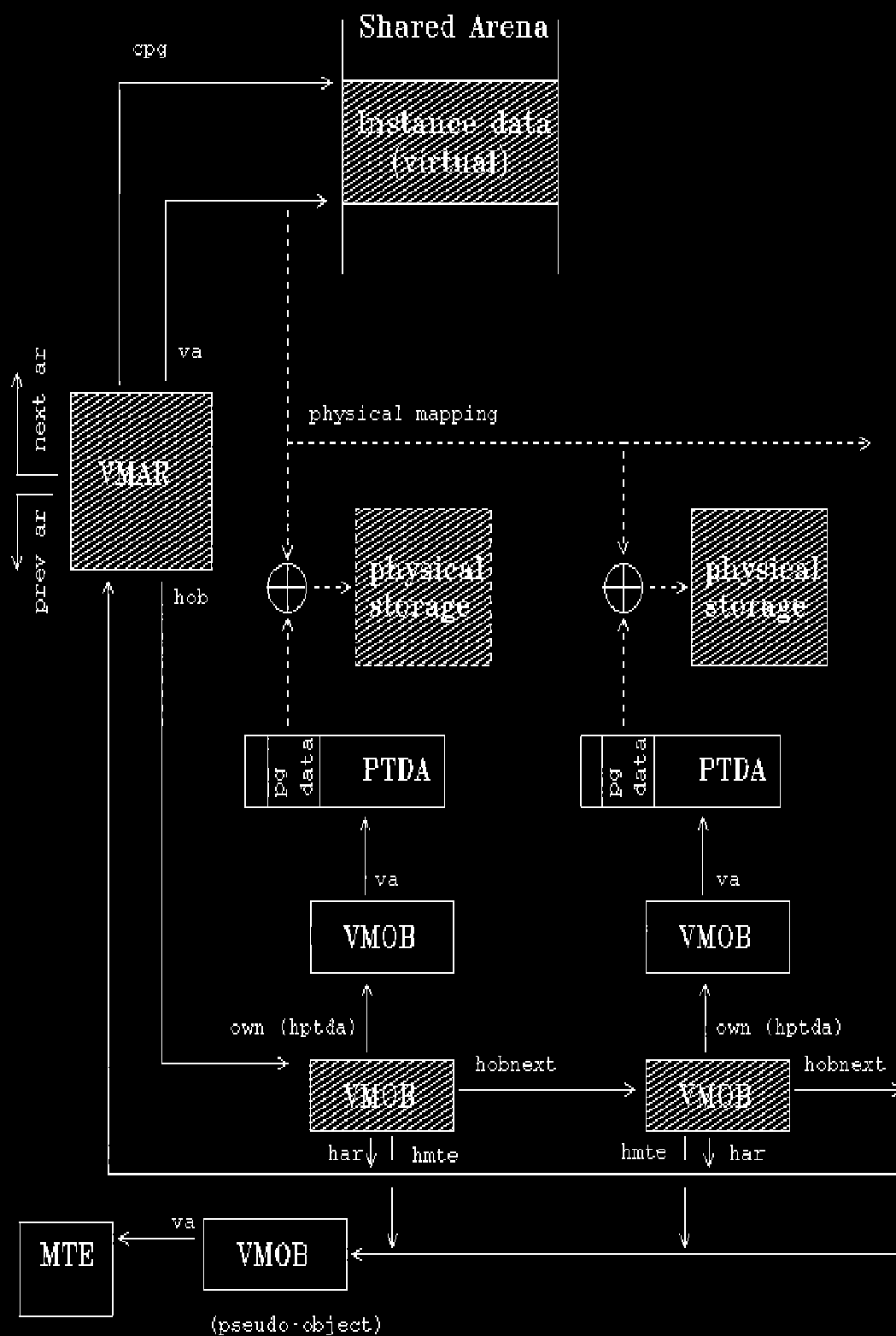
A DLL Instance data allocation shares only its address range among its accessors. The data is mapped to a different set of physical pages for each process.

This type of allocation is represented by a single VMAR with a chained list of VMOBs, one for each accessor. This is the only case where VMOBs are linked by the **hobnxt** field.

The following diagram depicts this situation.

Control blocks and data that directly represent the allocation are shown shaded.

# Shared Arena Instance Data



---

## The Page Frame Structure

Occasionally we need to enquire into the ownership and disposition of real storage. The PF is used to track the use of all frames of real storage, whether allocated, idle (pending freeing) or free.

The PF is formatted using the **.MP** KDB and DF command. .MP will optionally take the frame number (real address MOD 4K) as a parameter.

```
# .mp
ffefb000 InUse: pVP=ff406000 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00000
ffefb00c InUse: pVP=ff406050 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00001
ffefb018 InUse: pVP=ff40605a RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00002
ffefb024 InUse: pVP=ff406064 RefCnt=0002 Flg=0 ll=00 sl=00 Blk=00000 Frame=00003
ffefb030 InUse: pVP=ff40606e RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00004
```

Of particular interest are:

Frame=ffff

The real storage frame number represented by this PF.

pVP

The address of the related Virtual Page Structure (non-free PFs only). See next section.

ll

The long term lock count.

This is non-zero when an otherwise non-resident page is long-term locked, that is prohibited from being discarded or swapped, and expected to be so for a relatively long time.

sl

The short term lock count.

This is non-zero when an otherwise non-resident page is short-term locked, that is prohibited from being discarded or swapped, but temporarily so (much less than 10 seconds).

---

## The Virtual Page Structure

VPs track the disposition of every page of virtual storage of every object. They enable the system to locate the data for the page, whether it is in RAM or on the Swap file.

VPs are formatted using the **.MV** KDB and DF command, which takes as a parameter the address of the VP, which may be obtained from the PF structure.

```
.mv %ff4060f0 15
VPI=0018 pVP=ff4060f0 Res Frame=0011 Flg=410 HobPg=0001 Hob=0009 Ref=001
VPI=0019 pVP=ff4060fa Res Frame=0012 Flg=410 HobPg=0002 Hob=0009 Ref=001
VPI=001a pVP=ff406104 Res Frame=0013 Flg=410 HobPg=0003 Hob=0009 Ref=002
VPI=001b pVP=ff40610e Swp Block=08cc Flg=0a0 HobPg=027b Hob=0026 Ref=001
VPI=001c pVP=ff406118 Swp Block=0001 Flg=000 HobPg=0000 Hob=00e7 Ref=001
```

Of particular interest are:

Hob=nnnn

The hob of the object to which this page belongs.

HobPg=nnnn

The page number within the object.

Frame=ffff

The real storage frame number that backs this virtual page.

Block=bbbb

The Swap file 4K block that contains the data for this virtual page.

-----

## Page Management

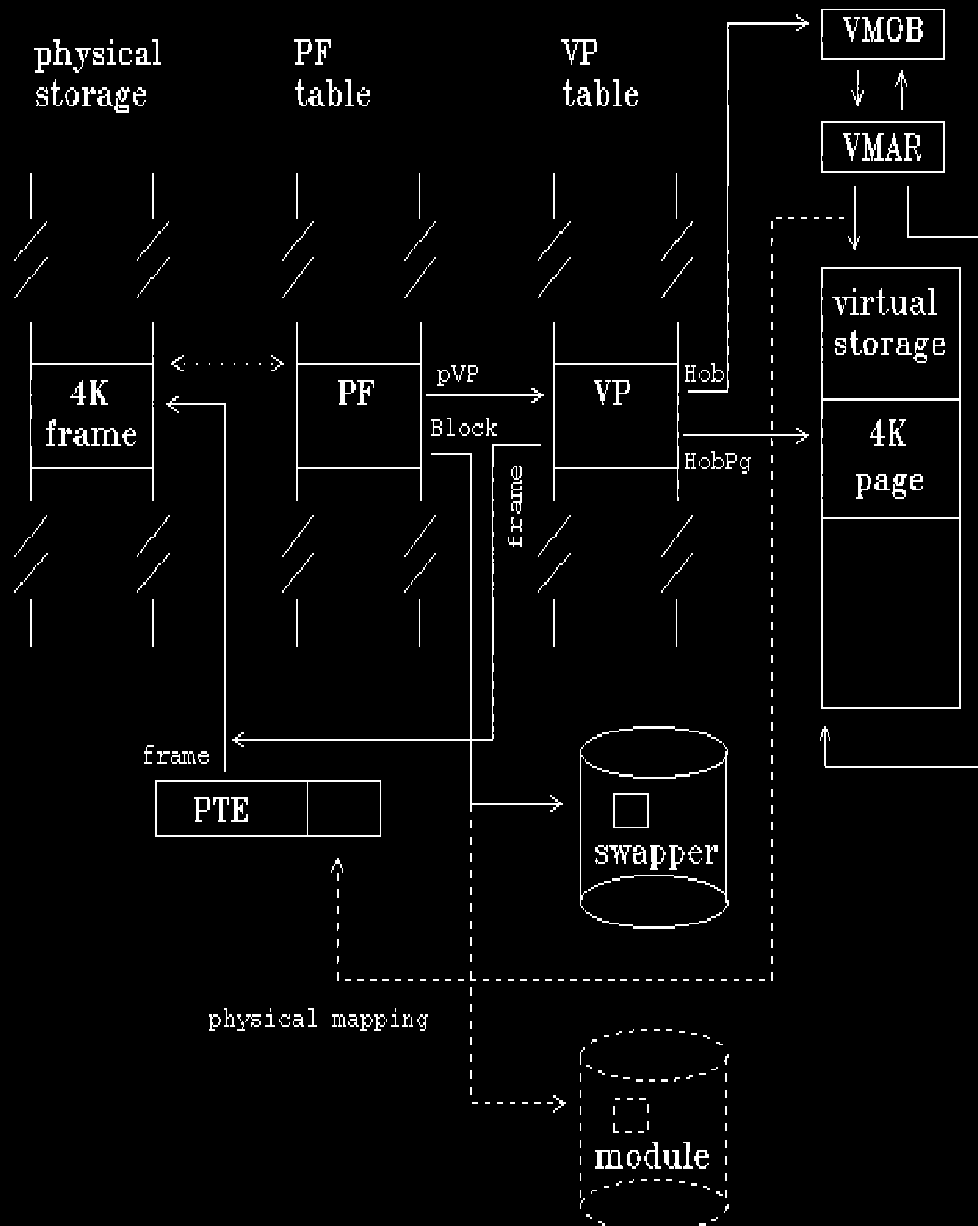
The relationship between **PF** structures, Page Table Entries, Swap File Blocks and Memory Objects is shown in the following diagrams.

The relationship between **PF** structures, **VP** structures, Page Table Entries, Swap File Blocks and Memory Objects is shown in the following two diagrams

The first of these depicts the situation where storage is backed or committed by physical memory.

# Page Management

## Backed Virtual Storage



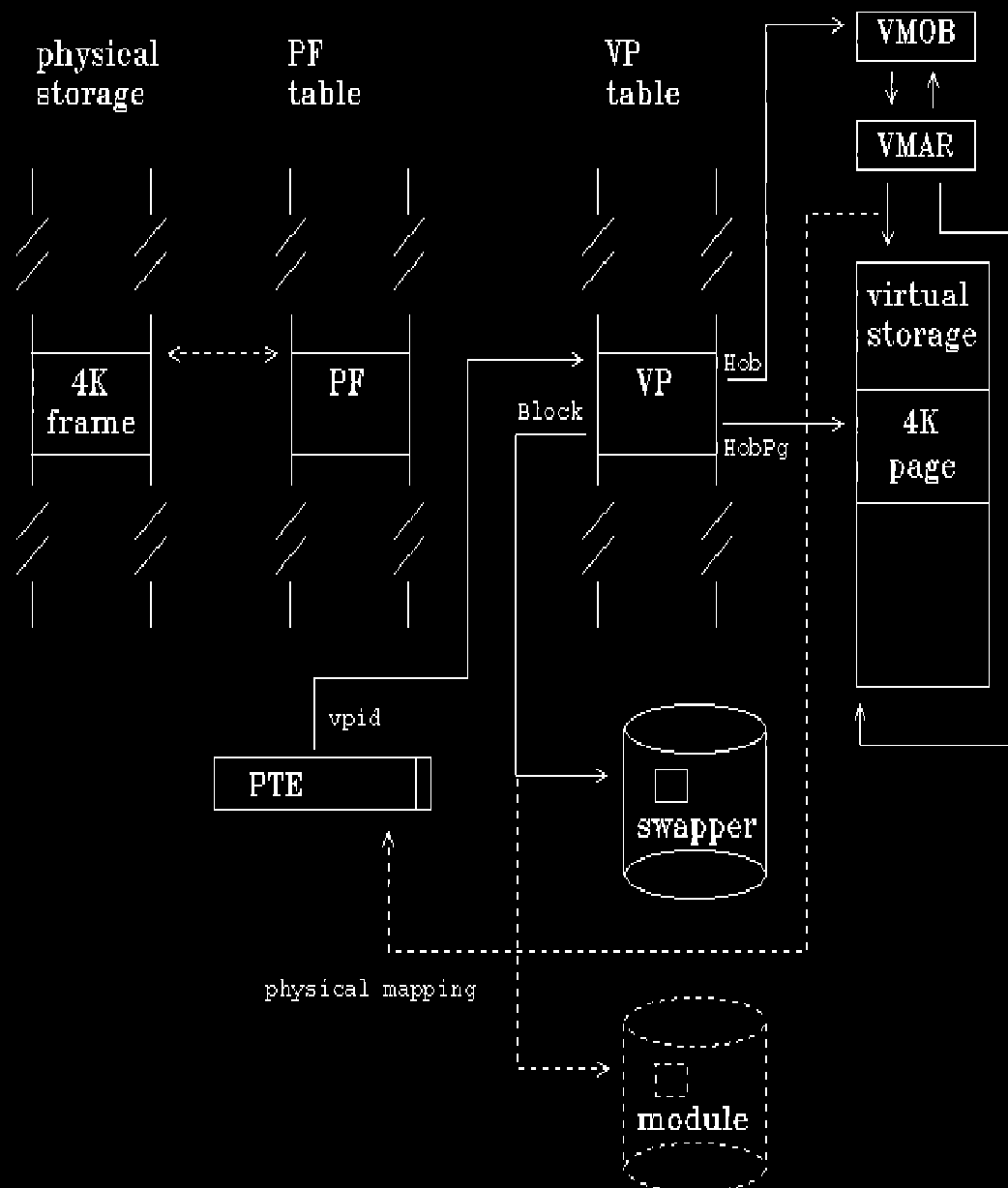
The next diagram shows how this situation changes when storage is paged out.

Note that the Page Table Entry is used to record the swapper block number when the page is not present.



# Page Management

## Unbacked Virtual Storage



---

# Aliasing

The situation described thus far can be further complicated by a technique known as aliasing. This is where one or more pages of an object may be mapped by page table manipulation to one or more pages of another object. In effect, this is partial object sharing.

This can occur between processes or within a process and is usually done for the following reasons:

A device driver needs to create an I/O buffer to receive data at interrupt time and therefore in any context. The application that called the device driver also needs to have access to the results. This is commonly solved by the device driver making a [UVIRT](#) allocation in the system arena which aliases an application data buffer.

A debugging application needs to access or even modify data and code of the debuggee. This is achieved through CS and DS selector aliasing.

A Dos Virtual Machine needs to simulate the A20 line wrap-around. Storage addressed above the A20 line aliases to addresses module 2\*\*20.

A Dos Virtual Machine's Private Arena address space is aliased in the system arena so that it may be accessed by Virtual Device Drivers in a context other than that of the VDM. The VDM handle (HVMD) is the alias address, which the VDD may add to any Private Arena Address to obtain a context independent access to a location in a given VDM.

These situations require the introduction of another memory management control block: the Alias Record (**VMAL**). Each VMAL has a unique handle or **hal**, which is the table entry from which the **VMAL** is allocated.

Where aliasing occurs, the handle to the alias record (**hal**) is saved in the VMAR of the aliasing address range.

In the case of memory aliasing the VMAL contains the handle to the PTDA of the aliasing process.

In the case of CS/DS aliasing within a process the VMAL contains the CS selector.

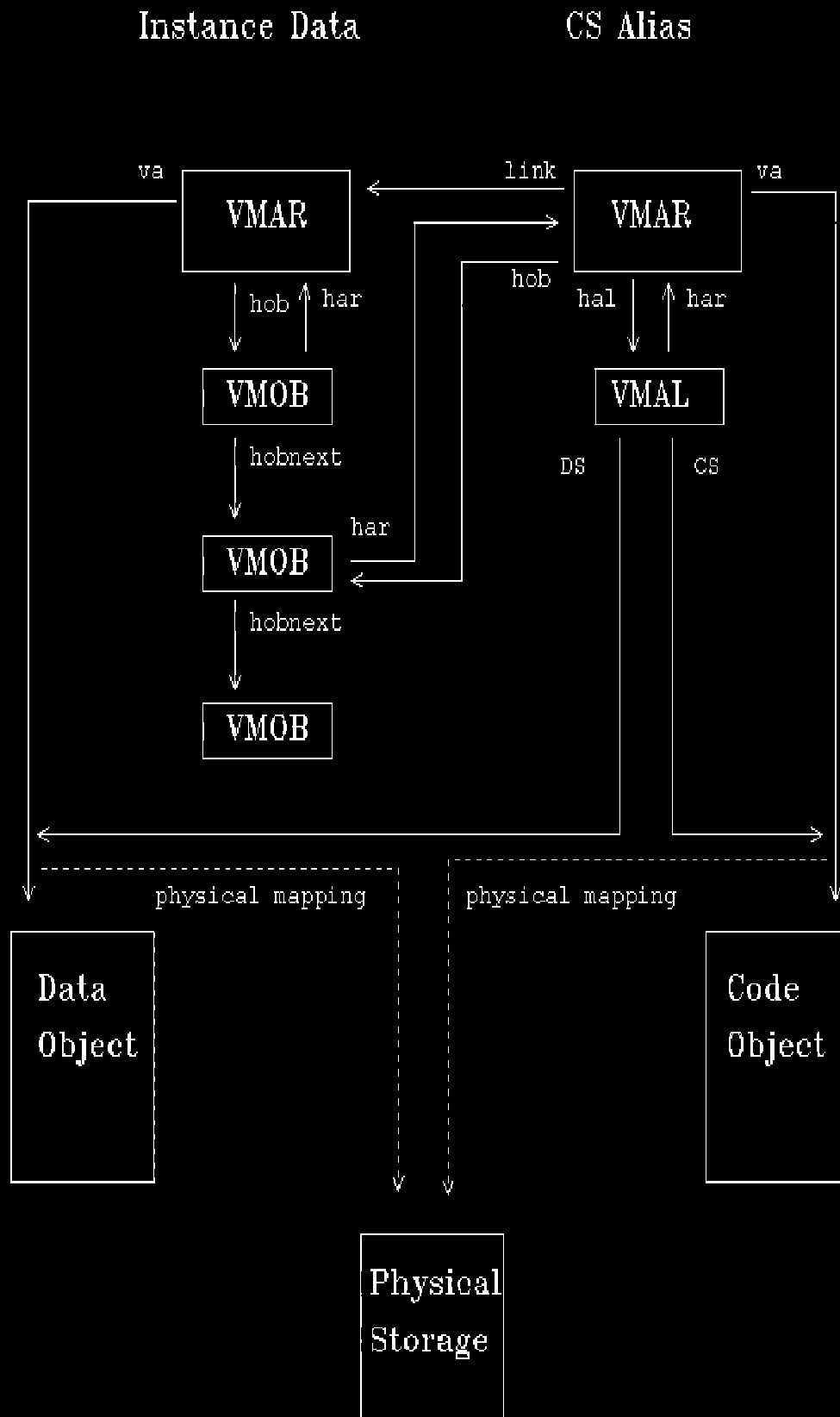
The link field of the VMAR is used to link together aliasing and aliased address ranges.

Alias records may be formatted using the **.ML** command as shown in the following example:

```
##.ml
hal=0001 pal=%fc5de020 har=00af hptda=00ae pgoff=00000 f=001
hal=0002 pal=%fc5de028 har=00b0 hptda=00ae pgoff=00000 f=001
hal=0003 pal=%fc5de030 har=007a hptda=00ae pgoff=00000 f=001
hal=0004 pal=%fc5de038 har=0160 cs=00e6 ds=d446 cref=001 f=13
hal=0005 pal=%fc5de040 har=017f hptda=00ae pgoff=00000 f=001
hal=0006 pal=%fc5de048 har=0197 hptda=00ae pgoff=00000 f=021
hal=0007 pal=%fc5de050 har=0198 hptda=00ae pgoff=00000 f=021
hal=0008 pal=%fc5de058 har=0199 hptda=00ae pgoff=00000 f=021
hal=0009 pal=%fc5de060 har=01c8 hptda=00ae pgoff=00000 f=001
hal=000a pal=%fc5de068 har=01db cs=0056 ds=d446 cref=001 f=13
hal=000b pal=%fc5de070 har=020b cs=0056 ds=d446 cref=001 f=13
hal=000c pal=%fc5de078 har=0242 cs=0056 ds=d446 cref=001 f=13
##
```

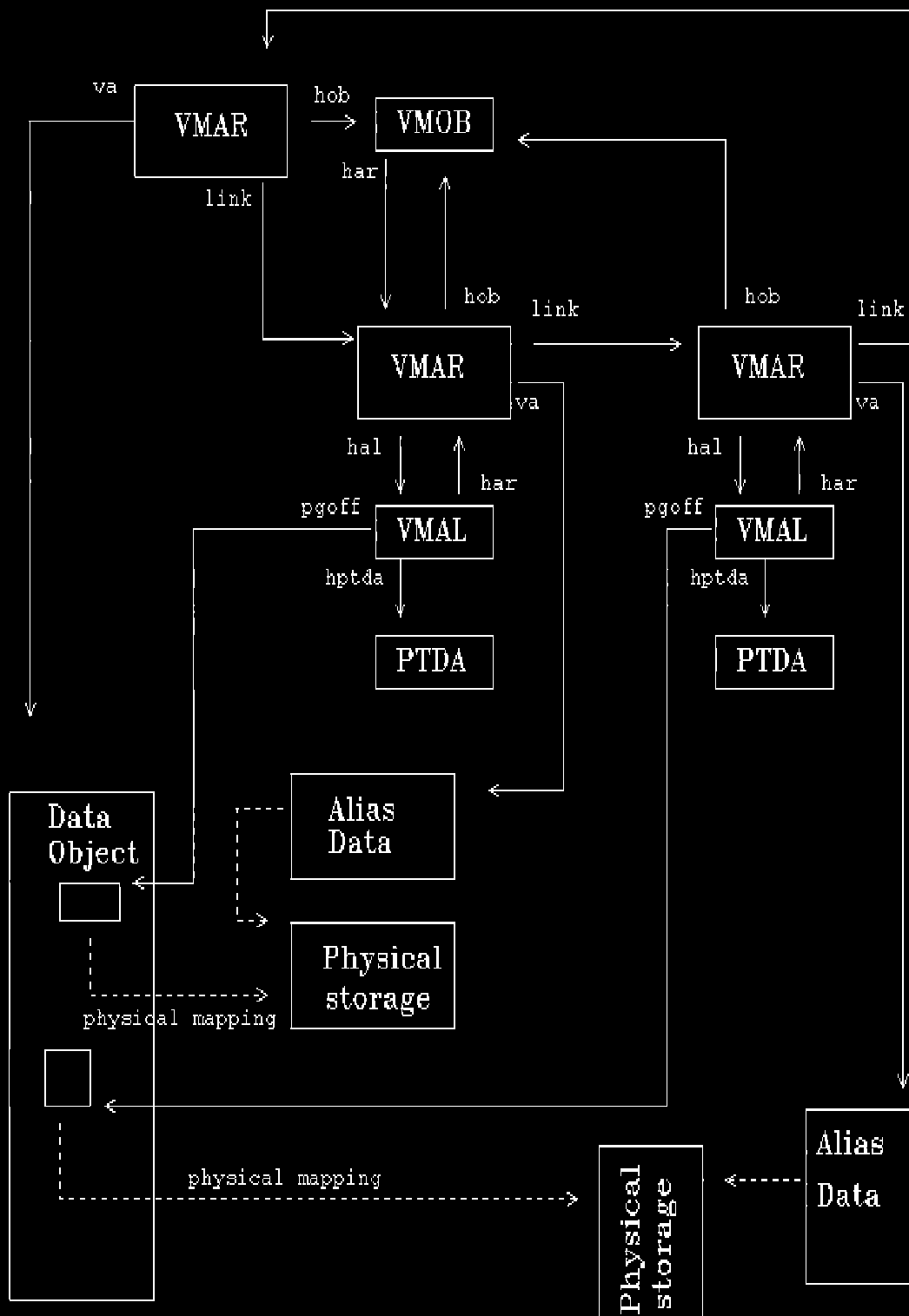
CS Aliasing is depicted in the following diagram:

# CS Alias of Shared Instance Data



The following diagram depicts multiple process memory aliasing:

# Memory Aliases in Multiple Processes



---

## Who Owns Virtual Memory?

Given a virtual address, the procedure for determining who owns and is using this memory essentially amounts to the following steps:

1. If the question of ownership relates to a known process's private storage then determine its hptda.
2. Locate the arena record(s) that encompass the address.
3. If more than one then select the one that relates to the process of interest (if known) by matching the hptda.
4. Locate the object records that are chained to the arena record.
5. If more than one then select the process of interest by matching the hptda.
6. Note the own and hmte values and their interpretation on the right-hand side of the VMOB display.
7. If necessary format the own and hmte VMOBs.
8. If either is an MTE then use .LM or .LMO, with the hob as parameter, to format the MTE.
9. If the memory is shared (hco=nnnnn appears in the arena record display) then format the chain of VMCOs and select the one that matches hptda from step 1.

Fortunately this task is reduced in complexity because of the **M** or match option that exists with both the .MO and .MA commands.

.MOM *addr* will display the VMOB of a pseudo-object that matches the *addr* if it exists. PTDA's are pseudo-objects and their addresses are listed by the .P command.

.MAM *addr* will search for all arena records whose address range encompasses *addr*. Under the kernel Debugger this search is restricted to the current context unless the **A** option (all contexts) is also specified. Under the dump formatter **A** is always in effect.

The **C** option further reduces the effort. This is the chain option and is applicable to .MO, .MA, .MC and .ML commands. Chaining formats all VMOBs, VMARs, VMCO and VMALs that are chained from each VMAR associated with the VM control block being formatted.

.MAMC (or .MAMAC under the DF) are the default options if just .M is specified. Furthermore the matching address defaults to the current CS:EIP.

The following sections illustrate memory ownership in:

Shared Arena Global Data

Shared Arena Instance Data

Private Arena Shared and Private Data

Physical Storage.

Further examples in memory management exploration, including looking at aliasing may be found in [Exploring Memory Management](#).

---

## Shared Global Data

### Who owns %12123456?

```
#.m %12123456
```

```
*har    par    cpg      va    flg next prev link hash hob    hal
0297 %feef2904 00000660 %11fb0000 369 0312 0295 0000 009e 02a1 0000 hco=0057f
hob    har hobjxt flgs own  hmte  sown,cnt lt st xf
02a1 0297 0000 4a2d fff5 0302 0000 00 00 00 00 give
hco=057f pco=ffe70b96 hconext=00241 hptda=06d1 f=1e pid=0059
hco=0241 pco=ffe6fb60 hconext=004ce hptda=04b2 f=1e pid=0043 c:pmspool.exe
```

```

hco=04ce pco=ffe70821 hconext=0034c hptda=05c3 f=1e pid=0016 c:pawn.exe
hco=034c pco=ffe70097 hconext=001e4 hptda=04ca f=1e pid=000f c:pmsshell.exe
hco=04ca pco=ffe7080d hconext=00348 hptda=05c3 f=16 pid=0016 c:pawn.exe
hco=0348 pco=ffe70083 hconext=0017a hptda=04ca f=16 pid=000f c:pmsshell.exe
hco=017a pco=ffe6f77d hconext=00177 hptda=03d9 f=16 pid=000b c:spdaemon.exe
hco=0177 pco=ffe6f76e hconext=00148 hptda=03ec f=16 pid=000c
hco=0148 pco=ffe6f683 hconext=000b2 hptda=03c9 f=16 pid=000a c:ddaemon.exe
hco=00b2 pco=ffe6f395 hconext=00083 hptda=0359 f=16 pid=0009 c:harderr.exe
hco=0083 pco=ffe6f2aa hconext=00081 hptda=02e1 f=16 pid=0007 c:landll.exe
hco=0081 pco=ffe6f2a0 hconext=0007d hptda=02ad f=16 pid=0006 c:lanmsgex.exe
hco=007d pco=ffe6f28c hconext=00037 hptda=027a f=16 pid=0008 c:pmsshell.exe
hco=0037 pco=ffe6f12e hconext=00000 hptda=02ac f=16 pid=0004 c:gambit.exe

```

```

# .mo 302
hob      va      flgs own  hmte   sown,cnt lt st xf
0302    %fdf40844 8000 ffa6 0000 0000 00 00 00 00 mte      c:pmmmerge.dll
#

```

This is shared arena global data because of the presence of the hco chain. The storage was dynamically allocated by PMMERGE.DLL as giveable storage. It is currently being referenced by 14 processes.

## Shared Instance Data

### Who allocated %13fa1234?

```

# .m %13fa1234
*har      par      cpq      va      flg next prev link hash hob      hal
009c    %feeeefd72 000000010 %13fa0000 179 009b 009d 0000 0000 063f 0000      =0000
hob      har hobnxt flgs own  hmte   sown,cnt lt st xf
063f    009c 0497    002c 06d1 00a3 0000 00 00 00 00 priv 0059
0497    009c 05c4    002c 04b2 00a3 0000 00 00 00 00 priv 0043 c:pmspool.exe
05c4    009c 04ce    002c 05c3 00a3 0000 00 00 00 00 priv 0016 c:pawn.exe
04ce    009c 03f0    002c 04ca 00a3 0000 00 00 00 00 priv 000f c:pmsshell.exe
03f0    009c 03dd    002c 03ec 00a3 0000 00 00 00 00 priv 000c
03dd    009c 03cd    002c 03d9 00a3 0000 00 00 00 00 priv 000b c:spdaemon.exe
03cd    009c 035d    002c 03c9 00a3 0000 00 00 00 00 priv 000a c:ddaemon.exe
035d    009c 02f4    002c 0359 00a3 0000 00 00 00 00 priv 0009 c:harderr.exe
02f4    009c 02e5    002c 027a 00a3 0000 00 00 00 00 priv 0008 c:pmsshell.exe
02e5    009c 02ae    002c 02e1 00a3 0000 00 00 00 00 priv 0007 c:landll.exe
02ae    009c 02a8    002c 02ad 00a3 0000 00 00 00 00 priv 0006 c:lanmsgex.exe
02a8    009c 0000    002c 02ac 00a3 0000 00 00 00 00 priv 0004 c:gambit.exe

```

```

# .mo a3
hob      va      flgs own  hmte   sown,cnt lt st xf
00a3    %fdef5f70 8000 ffa6 0006 0000 00 00 00 00 mte      c:doscall11.dll

```

```

# .lmo a3
hmte=00a3 pmte=%fdef5f70 mflags=0498b594 c:\os2\dll\doscall11.dll
obj      vsize     vbase     flags     ipagemap cpagemap hob sel
0001 00000360 1bf80000 80009025 00000001 00000001 00ac dfc6 r-x shr alias iopl
0002 0000aa34 1bf90000 80002025 00000002 0000000b 00ab dfcf r-x shr big
0003 0000d499 1bfa0000 8000d025 0000000d 0000000e 00aa dfd6 r-x shr alias conf iopl
0004 0000d4f0 1bfb0000 8000d025 0000001b 0000000e 00a9 dfde r-x shr alias conf iopl
0005 00001140 13f90000 80001023 00000029 00000002 00a8 9fcf rw- shr alias
0006 00001af0 13fa0000 80001003 0000002b 00000002 0000 9fd7 rw- alias
0007 00000e44 13fb0000 80001023 0000002d 00000001 00a6 9fdf rw- shr alias
0008 00000550 13fc0000 80001003 0000002e 00000001 0000 9fe7 rw- alias
0009 00001000 13fd0000 80001023 0000002f 00000000 00a4 9fef rw- shr alias
000a 00001000 13fe0000 80001023 0000002f 00000000 00a2 9ff7 rw- shr alias
#

```

%13fa1234 is the shared arena (there is no hptda associated with the arena record).

This is shared instance data because of the chain of related object records.

The storage was allocated by hmte=a3, but there are multiple owners.

mte a3 is DOSCALL1.DLL

%13fa1234 is within object 6 of the module. It is DLL RW instance data.

## Private Data

### Who owns #17:0 in thread slots 8 and 9?

>> First find the hptda's for each of the slots of interest since we are  
>> looking at private arena storage

```
# .p8
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
0008  0008 0001 0008 0007 blk 0200 abd2f000 abe497f0 abe28bf0      01 PMSHL32
# .mom %abe497f0
hob      va      flgs own  hmte  sown,cnt lt st xf
027a  %abe497f0  8000 ffc b ff79  0000 00  00 00 00 ptda 0008 c:pmsshell.exe

# .p 9
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
0009  0004 0001 0003 0001 blk 081f abd30000 abe48614 abe28de8      00 GAMBIT
# .mom %abe48614
hob      va      flgs own  hmte  sown,cnt lt st xf
02ac  %abe48614  8000 ffc b 02a8  0000 00  00 00 00 ptda 0004 c:gambit.exe
```

>> Next list all the owners of 17:0

```
# .m #17:0
*har      par      cpg      va      flg next prev link hash hob      hal
026d  %feef2568 00000010 %00020000 1d9 029a 026c 0000 0000 029d 0000 hptda=02ad
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
029d  026d 0000  0838 029e 029e  0000 00  00 00 00 shared      c:lanmsgex.exe

*har      par      cpg      va      flg next prev link hash hob      hal
0277  %feef2644 00000010 %00020000 1d9 0276 0272 0000 0000 02b0 0000 hptda=02ac
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
02b0  0277 0000  0838 02b1 02b1  0000 00  00 00 00 shared      c:gambit.exe

*har      par      cpg      va      flg next prev link hash hob      hal
02a0  %feef29ca 00000010 %00020000 179 02a4 029f 0000 0000 02e8 0000 hptda=02e1
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
02e8  02a0 0000  002c 02e1 02e7  0000 00  00 00 00 priv 0007 c:landll.exe

*har      par      cpg      va      flg next prev link hash hob      hal
02aa  %feef2aa6 00000010 %00020000 179 02ab 02a9 0000 0000 02f8 0000 hptda=027a
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
02f8  02aa 0000  002c 027a 02f7  0000 00  00 00 00 priv 0008 c:pmsshell.exe

*har      par      cpg      va      flg next prev link hash hob      hal
02fc  %feef31b2 00000010 %00020000 1d9 02fd 02fb 0000 0000 0360 0000 hptda=0359
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0360  02fc 0000  0838 035f 035f  0000 00  00 00 00 shared      c:harderr.exe

*har      par      cpg      va      flg next prev link hash hob      hal
0360  %feef3a4a 00000010 %00020000 1d9 0361 035f 0000 0000 03d0 0000 hptda=03c9
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
03d0  0360 0000  0838 03cf 03cf  0000 00  00 00 00 shared      c:ddaemon.exe

*har      par      cpg      va      flg next prev link hash hob      hal
036b  %feef3b3c 00000010 %00020000 1d9 036c 036a 0000 0000 03e0 0000 hptda=03d9
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
03e0  036b 0000  0838 03df 03df  0000 00  00 00 00 shared      c:spdaemon.exe

*har      par      cpg      va      flg next prev link hash hob      hal
0378  %feef3c5a 00000010 %00020000 1d9 0379 0377 0000 0000 03f3 0000 hptda=03ec
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
03f3  0378 0000  0838 03f2 03f2  0000 00  00 00 00 shared

*har      par      cpg      va      flg next prev link hash hob      hal
040e  %feef493e 00000010 %00020000 179 045c 040f 0000 0000 04c6 0000 hptda=04b2
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
04c6  040e 0000  002c 04b2 0522  0000 00  00 00 00 priv 0043 c:pmspool.exe
```



```

*har      par      cpgr      va      flg next prev link hash hob      hal
0427 %feef4b64 00000010 %00020000 179 0428 0426 0000 0000 04cf 0000 hptda=04ca
hob      har hobnxt flgs own hmte sown,cnt lt st xf
04cf 0427 0000 002c 04ca 02f7 0000 00 00 00 00 priv 000f c:pmshell.exe

*har      par      cpgr      va      flg next prev link hash hob      hal
04e8 %feef5bfa 00000010 %00020000 179 04e6 04e5 0000 0000 05d4 0000 hptda=05c3
hob      har hobnxt flgs own hmte sown,cnt lt st xf
05d4 04e8 0000 002c 05c3 05cf 0000 00 00 00 00 priv 0016 c:pawn.exe

*har      par      cpgr      va      flg next prev link hash hob      hal
0502 %feef5e36 00000010 %00020000 1d9 059f 0598 0000 0000 0507 0000 hptda=06d1
hob      har hobnxt flgs own hmte sown,cnt lt st xf
0507 0502 0000 0838 05b3 05b3 0000 00 00 00 00 shared

*har      par      cpgr      va      flg next prev link hash hob      hal
0507 %feef5ea4 00000010 %00100000 1e1 056c 05cb 05d4 0000 0678 0018 hptda=04af
hal=0018 pal=%fddae0d8 har=0507 hptda=04af pgoff=00000 f=081
har      par      cpgr      va      flg next prev link hash hob      hal
05d4 %feef7042 00000040 %00000000 1e1 05bf 0461 0000 0000 0678 0000 hptda=04af
hob      har hobnxt flgs own hmte sown,cnt lt st xf
0678 0507 0000 103c 04af 0000 0000 00 00 00 00 priv 005b *vdm

>> Slot 8:

# .mo 2b1
hob      va      flgs own hmte sown,cnt lt st xf
02b1 %feeeef38 8000 ffa6 02a7 0000 00 00 00 00 mte c:gambit.exe

# .lmo 2b1
hmte=02b1 mpte=%feeeef38 mflags=00003140 c:\dcaf13\gambit.exe
seg sect psiz vsiz hob sel flags
0001 0002 1fe0 1fe0 02b2 000f 2d20 code shr rel
0002 0013 002a 002c 02b0 0017 2d20 code shr rel
0003 0014 19ae 19ae 0000 001f 0d01 data rel
0004 0022 0002 0002 02a9 0027 2c20 code shr
0005 0000 0000 3400 0000 002f 0c01 data

#

>> Slot 9

# .mo 2f7
hob      va      flgs own hmte sown,cnt lt st xf
02f7 %fdf40a18 8000 ffa6 0000 0000 00 00 00 00 mte c:pmshell.exe
#

```

This is private arena data of some sort, whose address range is present in 13 processes.

The hptda for pid 4 (slot 9 is 2ac)

The second major entry from .m output (har=277, hptda=2ac) is for gambit.exe in pid 4.

The owner and hmte are the same (2b1). This indicates a code segment within the module gambit.exe.

.LMO 2b1 show this to be in segment 2 of gambit.exe

The storage in pid 8 (slot 8) is shown in the 4th entry, har=2aa.

Here own=27a and hmte=2f7.

The owner is shown to the right of the VMOB as being pid 8. We can check this by displaying hob 27a. This turns out to be a ptda for pid 8, as we saw when we used .mom against the PTDA address.

.lmo 2f7 shows this to be the MTE for pmshell.exe. We concluded that pmshell has allocated private memory in pid 8 at this address.

-----

## Physical Memory

### Who owns physical address %%90123?

```
# .mp 90
ffefb6c0 InUse: pVP=ff408576 RefCnt=0001 Flg=1 ll=00 sl=00 Blk=00272 Frame=00090

# .mv %ff408576
VPI=03bf pVP=ff408576 Swp Frame=0090 Flg=030 HobPg=0011 Hob=0282 Ref=001

# .moc 282

*har      par      cpg      va      flg next prev link hash hob      hal
0263 %feef248c 00000060 %a8650000 001 0262 0267 0000 0001 0282 0000      =0000
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0282  0263 0000  0000 ffd4 0000  0000 00  00 00 00 vddheap

# dp %a8650000+11000 ll
linaddr  frame  pteframe  state res Dc Au CD WT Us rW Pn state
%a8661000* 0055d  frame=00090 0    0  c  A          s  r  P  pageable
%a8661000 00090  frame=00090 0    0  c  A          s  r  P  pageable
#
```

Physical address %%90123 is in frame 90.

This is currently assigned to VP at %ff408576 and is on the swap file at block 272.

The VP tells us the hob and page within the hob.

.MOC will format the VMOB and associated VMAR.

We can check that this is correct from the page table entries for the 17th (0x11th) page of the object's virtual address.

-----

## Thread Scheduling and Dispatching Topics

Part 2 of our discussion on The Wait Condition centres on the Scheduler and Dispatcher and the mechanisms that govern when threads will or will not run.

This is considered from the perspective the system, which leads us to divide the discussion into two cases:

[Blocking - Voluntary Suspension](#)

[Involuntary Suspension](#)

-----

## Blocking - Voluntary Suspension

We now turn our attention to *blocking*, which is the mechanism that threads use to give up processor time voluntarily to wait for an event to occur or a resource to become available.

The term voluntary is chosen from the perspective of the scheduler and not necessarily from the application's perspective. In this context voluntary suspension refers to an action taken by a thread to give up its time-slice. This will include direct actions such as waiting on semaphores as well as calling APIs, which for internal reasons need to wait for a resource or an event.

PROCLOCK and its counterpart PROCRUN are the two kernel routines at the heart of the block/run mechanism. These are callable directly by kernel component and also by Device Drivers and File System Drivers through a small interface layer. Application code only gets to call PROCLOCK and PROCRUN indirectly through system APIs and in particular through the semaphore APIs.

The block/run mechanism is designed with the following criteria:

A thread should be able to block without the waking thread having to know whether anyone, or who, had blocked on a resource

Multiple threads should be able to wake when an event or resource becomes available.

This is achieved by having an abstract token, known as the **Block ID**, associated with the resource or event. The BlockId is passed to PROCBLOCK when a thread blocks. Similarly when another thread wishes to wake threads waiting for a resource or event the BlockId that represents the resource or event is passed to PROCRUN.

In addition to the BlockId, callers of PROCRUN receive a flag that indicates whether all or just the highest priority thread waiting on the BlockId should wake.

This mechanism has shortcomings unless certain constraints are applied:

BlockIds need to be subject to a convention that gives uniqueness otherwise it is possible that threads will incorrectly block and run. A solution is to use the address of a control block memory object that relates uniquely to the resource or event.

If addresses are to be used for BlockIds then they must point to global data for reasons of uniqueness. Furthermore, if they are to be reference by disabled code then the storage needs to be in resident memory. This more or less implies that addresses must be taken from within the System Arena.

If BlockIds are in use that do not represent addresses then they must not conflict with any potential addresses used as BlockIds.

Even if addresses are use there is no accounting information that says who owns the related resource.

A workable scheme is implemented by limiting the direct use of PROCBLOCK and PROCRUN to system code, device drivers and file system drivers, all of which have access to the System Arena.

Apart from three special conventions the system and most device drivers use addresses as BlockIds. There are three system defined conventional BlockIds are:

ffe....	results from a RAMSEM wait.
ffd....	results from a MUXWAIT.
ffca....	results from a Child Wait
x..... (x=a - f)	Linear address of the memory object of control block that relates to the resource.
.....	Probably selector:offset address of the memory object or control block that relates to the resource.

This scheme could be subverted by device drivers, but in general they will choose to block on addresses of resources they own, which are usually allocated out of the system arena and addressed using a GDT select:offset.

Accountability remains an exposure. For BlockIds that are addresses the owner of the memory that the BlockId points to gives a big clue. For conventional BlockIds we have to do more work. These are discussed in detail later. We will first we look at an example of a BlockId that is an address.

#### **Basic Technique:**

The technique for analysing blocked threads is two-pronged:

1. We can look at the wait from the application perspective by examining the current user registers and by trying to identify the API issued. This is usually relatively easy but often gives no clue as to the underlying wait since any single API may block on many occasions for many reasons.
2. Examine the problem from the Internal, or Kernel perspective to determine what an API might be waiting for. This process starts with finding the associated BlockId.

When a thread blocks its BlockId is stored in the TCB TCBSleepId field. Conveniently, this is formatted by using the **.PB** KDB and DF command.

#### **Note:**

.PB under DF lists non-blocked threads. BlockIds are irrelevant for such threads.

.PB also attempts to interpret the BlockId. The full details of these are given in the [Kernel Debugger and Dump Formatter Command Reference](#). In addition to classifying the BlockId, .PB examines TCB\_SemInfo and TCB\_SemDebugAddr.

For many semaphore originated BlockIds TCB\_SemInfo is used to store the address or handle of the user's semaphore that lead to the

thread blocking. .PB will attempt to locate a near symbol to the semaphore address and display it.

Under the kernel Debugger, TCB\_SemDebugAddr is used to store the address of the caller to the Semaphore API when the thread blocked. If this field is not 0xffffffff .PB attempts to locate a near symbol to the caller and display it.

Once we have the BlockId, TCB\_SemInfo, and TCB\_SemDebugAddr we are able to begin searching for information associated with reason for blocking.

The next step is to decide whether the BlockId is one of the three special categories or to be treated as an address.

## Blocking on the Address of a Resource

The initial analysis of BlockIds that are linear addresses uses the .M command to determine ownership.

If we have appropriate symbols loaded, LN against the BlockId can also be very informative.

As mentioned in the previous section, for addresses to be effective BlockIds they must be unique and so are generally allocated from the system arena. Most allocations from the system arena are 'labelled' with a system object Id. If the .MO command is used against a system object Id it will display a meaningful mnemonic for the Owner Id. In many cases the Mnemonic is for a system control block or buffer. BlockIds that address the beginning of a control block tend to be used for serialising updates to the control block. There may be processes that a control block is associated with. These are often serialised by using the address of a field within the control, that is specifically associated with the process.

A complete list of system object Ids may be found in the under the Kernel Debugger Command Reference under the [.MO command description](#).

We now look at some examples:

### File System - Device Driver

```
# .pb41
Slot Sta BlockID Name      Type      Addr      Symbol
0041 blk 04085ca7 DEMO1
# ln 408:5ca7
No Symbols Found
# .m 408:5ca7

*har      par      cpq      va      flg next prev link hash hob      hal
0079 %fe1fa70 00000010 %7bf27000 129 0078 0077 0000 0000 007b 0000      sel=0408
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
007b 0079 0000 0324 ffa1 0000 0000 00 00 00 00 sft

# .d sft 408:5ca7
      sf_ref_count: 0000                      sfi_mode: 0092
      sf_usercnt: 0000                        sfi_hVPB: 0000
      reserved: 00                          sfi_ctime: 0000
      sf_flags(2): 02c0:0000                  sfi_cdate: 0000
      sf_devptr: #0928:001c                   sfi_atime: 0000
      sf_FSC: #00c8:ff40                      sfi_adata: 0000
      sf_chain: #0000:0000                    sfi_mtime: 3cel
      sf_MFT: ffffffff                       sfi_mdate: 1eb0
sfdFAT_firFILEclus: 0000                      sfi_size: 00000000
sfdFAT_cluspos: 0000                          sfi_position: 000013c0
sfdFAT_lstclus: 0000                          sfi_UID: 0000
sfdFAT_dirsec: 00000000                      sfi_PID: 0012
sfdFAT_dirpos: 00                          sfi_PDB: 0000
sfdFAT_name: DEMODEV2                        sfi_selfsfn: 00b5
sfdFAT_EAHandle: 0000                        sfi_tstamp: 00
      sf_plock: 0000                          sfi_DOSattr: 00
      sf_NmPipeSfn: 0000
      sf_codepage: 0000

# .p41
Slot Pid Ppid Csid Ord Sta Pri pTSD      pPTDA      pTCB      Disp SG Name
0041 0012 000f 0012 0001 blk 0300 7bd19000 7bdfc218 7bddfc68 0ebc 13 DEMO1

# .s41
Current slot number: 0041

# .r
eax=00000000 ebx=00000002 ecx=00000000 edx=4d3409ea esi=d02f0021 edi=000009ea
```

```

eip=00000134 esp=0000a424 ebp=0000a43e iopl=2 -- -- -- nv up ei pl nz ac po nc
cs=000f ss=005f ds=005f es=004f fs=150b gs=0000 cr2=00000000 cr3=001ac000
000f:00000134 8946fe      mov     word ptr [bp-02],ax      ss:a43c=0d16

# u cs:ip-20
000f:00000114 681f00      push    001f
000f:00000117 682d00      push    002d
000f:0000011a 0e          push    cs
000f:0000011b e87c19      call    1a9a
000f:0000011e 83c40a      add     sp,+0a
000f:00000121 8e06ee09    mov     es,word ptr [09ee]
000f:00000125 8b5ef8      mov     bx,word ptr [bp-08]
000f:00000128 d1e3        shl     bx,1
000f:0000012a 26ffb764d2  push    word ptr es:[bx+d264]
000f:0000012f 9a0000ab1d  call    ldab:0000
000f:00000134 8946fe      mov     word ptr [bp-02],ax
000f:00000137 8e06ee09    mov     es,word ptr [09ee]

# dg ldab
ldab  CallG32 Sel:Off=0148:00004414      DPL=3 P  DWC=1

# ln 148:4414
0148:00004414 OS2KRNL DOSCLOSE

```

Slot 41 is waiting on BlockId 04085ca7. This is too low to be a linear address. We assume selector:offset.

.M 408:5ca7 reveals the owner to be **sft**. This is a System File Table structure.

The .D command will format STFs, so we do so using the BlockId as the SFT address.

This SFT represents a device driver called **DEMODEV2**. We can tell because there is no MFT pointer in the SFT and the flags indicate a device.

From the application side we unassemble back from the CS:IP.

The application has just issued a call-gate instruction.

Examination of the call-gate GDT descriptor show we were calling DOSCLOSE in the kernel.

We are waiting for the close to complete, possibly the device driver has not returned completion status to the last I/O request.

### Named Pipes

```

# .s 18
Current slot number: 0018
# .pb#
Slot Sta BlockID Name      Type      Addr      Symbol
0018# blk 06700012 EPWPSI
#

# .m 670:12

*har      par      cpg      va      flg next prev link hash hob      hal
00a8 %fef1fe7a 00000010 %7b563000 129 00a7 00a9 0000 0000 00b4 0000      sel=0670
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
00b4 00a8 0000 0124 ff31 0000 0000 00 00 00 00 npipenp

# .p#
Slot Pid Ppid Csid Ord Sta Pri pTSD      pPTDA      pTCB      Disp SG Name
0018# 000c 0000 000c 0001 blk 0200 7bc70000 7bd79964 7bd5b5f0 0ec8 00 EPWPSI

# .r
eax=00000000 ebx=00005552 ecx=00050000 edx=0000f020 esi=00000446 edi=00000302
eip=000014bb esp=000063da ebp=000063de iopl=2 -- -- -- nv up ei pl nz na po nc
cs=d01f ss=002f ds=beb7 es=0077 fs=150b gs=0000 cr2=00000000 cr3=001ac000
d01f:000014bb c9          leave
# u.cs:ip-10
Expression error
# u cs:ip-10
d01f:000014ab c9          leave
d01f:000014ac ca0a00      retf     000a
DOSCALL1 DOSCONNECTNPIPE:
d01f:000014af c8000000      enter   0000,00
d01f:000014b3 ff7606      push    word ptr [bp+06]
d01f:000014b6 9a0000131c    call    1c13:0000
d01f:000014bb c9          leave

```

```

d01f:000014bc ca0200      retf      0002
DOSCALL1 DOSDISCONNECTNPIPE:
d01f:000014bf c8000000    enter    0000,00
d01f:000014c3 ff7606      push     word ptr [bp+06]
d01f:000014c6 9a00001b1c   call    1c1b:0000
d01f:000014cb c9          leave   0000
d01f:000014cc ca0200      retf      0002

# dg 1c13
1c13 CallG32 Sel:Off=0148:0000540c      DPL=3 P   DWC=1
# ln 148:540c
0148:0000540c OS2KRNL DOSCONNECTNPIPE
#

```

In this example the BlockId is **06700012**. This is unlikely to be a linear address. We assume that it is **670:12**.

.M 670:12 shows the owner to be **npipenm**. This is a named pipe name segment. Could the process be waiting for a pipe connection?

Looking at the application side we see that the last ring 3 instruction to be executed was a call-gate, which turns out to be DOSCONNECTNPIPE in the Kernel.

These last two examples were reasonably revealing. More often than not we use .M against a BlockId ( and other system data) and we get one of:

```

vmkshrw
vmkshro
vmkrhrw
vmkrhro

```

These are, so called Public Kernel Heaps. Fortunately each allocated heap block is imbedded in a structure that reveals the owner of the block. This is discussed next.

## Kernel Public Heaps

The kernel has 7 heaps for general use by itself, device drivers and file system drivers. They have the following object id mnemonics:

<b>vmkshro</b>	Swappable read-only heap
<b>vmkshrw</b>	Swappable read/write heap
<b>vmkrhro</b>	Resident read-only heap
<b>vmkrhrw</b>	Resident read/write heap
<b>krhro1m</b>	Resident read-only < 1Mb heap
<b>krhrw1m</b>	Resident read/write < 1Mb heap
<b>kbdsym</b>	Resident kernel debugger symbol heap

Not all heaps are always built. Note in particular:

**hdbsym** is not present under the RETAIL kernel.

**vmkshrw** is used for the **krhro1m**, **krhrw1m**, **vmkshrw** and **vmkshro** heaps under the RETAIL kernel.

**vmkrhrw** is used for both **vmkrhrw** and **vmkrhro** under the RETAIL kernel.

Notice that each of the heaps is either resident or swappable.

Each heap is partitioned into blocks.

Swappable heap blocks have an 8 byte prefix followed by the block data.

Resident heap blocks have two forms:

Regular: for smaller allocation. These have a 4 byte prefix.

Attributed or extended: These use a 4 byte prefix and an 8 byte suffix.

-----

## Swappable Heap Blocks

Kernel swappable heap blocks for allocated blocks have the following layout:

<size><owner><selector><data>

Field	Bits	Description
size in bytes	63-32	Size of the block including the header in bytes ORed with signature 0x52000000.
owner	31-16	Owner of heap block. This is either a system owner (value between 0xff2d and 0xffff8, or a memory handle/pseudo handle such as an MTE pseudo-handle.
selector	15-0	GDT selector mapping block's data else null.

### Finding the owner of a Swappable Head Selector

```
# .m 8f0:0
```

```
*har      par      cpq      va      flg next prev link hash hob      hal
0021 %fef1f2e0 00001400 %fca5f000 121 0020 0022 0000 0020 0022 0000 =0000
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0022 0021 0000 0225 ffef 0000 0000 00 04 00 00 vmkshrw
```

```
# dl 8f0
```

```
GDT
```

```
08f0 Code Bas=fca95000 Lim=00008ed3 DPL=0 P RE A
```

```
# dd %fca95000-10
```

```
%fca94ff0 00000000 00000000 52008ee0 08f0ff49
%fca95000 08e8b81e 32b8d88e 16ca1f00 06c89000
%fca95010 1e560000 8e08e8b8 a23e83d8 06740009
%fca95020 eb63a5e8 c02b9003 0bfe4689 e90374c0
%fca95030 468b017e 10568b0e 52000805 6aff6a50
%fca95040 13969aff 5f3d1000 c4e77400 83260e5e
%fca95050 74000e7f 0142e903 0c47ff26 261276c4
%fca95060 2616448b 8918548b 5689fa46 0e468bfc
```

```
# .mo ff49
```

```
ff49 fsd2
```

```
# .lml
```

```
hmte=0982 pmte=%fe0e1a14 mflags=0408b186 e:\ibmlan\netlib\splla.dll
hmte=097e pmte=%fe0e1a54 mflags=0408b186 e:\ibmlan\netlib\lrhml.dll
hmte=0979 pmte=%fe0e1bac mflags=0408b186 e:\ibmlan\netlib\lrns1.dll
hmte=096b pmte=%fe0e1d60 mflags=0408b186 e:\ibmlan\netlib\netibm.dll
hmte=0164 pmte=%fe02cc40 mflags=0498b1c8 e:\os2\dll\sysmono.fon
.
.
.
hmte=0181 pmte=%fe02ccb0 mflags=4498b1d5 e:\os2\dll\pmatm.dll
hmte=031b pmte=%fe02af18 mflags=0428alc9 e:\ibmlan\netprog\netwksta.200
hmte=0306 pmte=%fe059f90 mflags=0428alc9 e:\netware\nwifs.ifs
hmte=0160 pmte=%fe01ff4c mflags=0428alc9 d:\dataex2\iwsfsd2.ifs
hmte=0117 pmte=%fdf5df60 mflags=0428alc9 e:\os2\cdfs.ifs
hmte=00d2 pmte=%fdf53990 mflags=0428alc9 e:\os2\hpfs.ifs
```

```
# .lmo 117
hmte=0117 pmte=%fd5df60 mflags=0428a1c9 e:\os2\cdfs.ifs
seg sect psiz vsiz hob sel flags
0001 0002 8ed3 8ed4 0000 08f0 8d60 code shr prel rel
0002 004a 0964 0ad0 0000 08e8 8c41 data prel
#
```

We use .M to find that the owner of 8f0:0 is **vmkshrw**.

So, we look at the descriptor for 8f0 to find it's base address. Note that the selectors assigned to kernel heap blocks address the data portion only.

We dump out 0x10 bytes before the selector base to show the block header to be 0x52008ee0 0x08f0ff49. This tells us the length of the block including header is 8ee0. (Data sizes are rounded up to the next quad-word). The user of the block is ff49.

#### Note:

The following short cut could have been used:

```
dd %(8f0:0)-10
```

.MO ff49 shows **fsd2**. This is the second file system driver to initialise.

.LML will list DLLs, Fonts, and FSDs, newest first. Counting back from the end we see FSD1 is HPFS and FSD2 is CDFS.

.LMO 117 confirms that 8f0:0 does belong to CDFS.IFS.

## Resident Heap Blocks

Kernel resident heap blocks are of two types, regular and attributed.

Regular blocks are the simplest and most common type. They have the form:

```
<simple header><data>
```

<simple header> is a dword (32-bits) having the following layout

```
<owner><prev block free flag><size in dwords><yielded flag><type flag>
```

Field	Bits	Description
owner	31-16	Owner of heap block. This is either a system owner (value between 0xff2d and 0xff8, or a memory handle/pseudo handle such as an MTE pseudo-handle.
previous block free flag	15	1 if previous block is free, else 0
size in dwords	14-2	Size of the block including the header in dwords.
yielded flag	1	1 if a free block search yielded the CPU while looking at this block, else 0
type flag	0	0 (indicates Regular Block)

Extended blocks contain a two-part header and have the following form:

```
<size header><data><header extension>
```

<size header> is a dword (32-bits) having the following layout

```
<extra flags><size in dwords><yielded flag><type flag>.
```

Field	Bits	Description
-------	------	-------------



extra flags	31-24	Additional flags. Bit 31 - set if block is free Bit 30 - set if prev block is free Bits 29-24 - reserved and 0
size in dwords	23-2	Size of the block including the header in dwords.
yielded flag	1	1 if a free block search yielded the CPU while looking at this block, else 0
type flag	0	1 (indicates Extended Block)

<data> is the data area available for use by the client and is always dword granular and dword aligned.

<header extension> is a dword-granular structure containing the following information

<owner><selector><hmte><pad>

Field	Bits	Description
owner	63-48	Owner of heap block. This is either a system owner (value between 0xff2d and 0xffff8, or a memory handle/pseudo handle such as an MTE pseudo-handle.
selector	47-32	GDT selector mapping block's data else null.
hmte	31-16	hmte associated with this heap block?
pad	15-0	padding for double word alignment

When a block is free, its data portion contains additional information. The first two dwords contain forward and backward pointers to the next and previous blocks on the free list. The last dword contains a copy of the previous block pointer. Note that extended free blocks do not have an owner field, so bit 31 of their header is set indicating that they are free.

The **hmte** field of the header extension is no longer used for any specific purpose.

#### Now for an example of a regular heap block.

```
# .s 47
Current slot number: 0047

# .pb #
Slot Sta BlockID Name Type Addr Symbol
0047# blk fe04c8e8 PMSHL32

# .m %fe04c8e8

*har par cpg va flg next prev link hash hob hal
0003 %fef1f04c 00001000 %fdf1f000 001 0002 0020 0000 0000 0003 0000 =0000
hob har hobnxt flgs own hmte sown,cnt lt st xf
0003 0003 ff08 0000 ffec 0000 0000 00 06 00 00 vmkrhrw
# dd %fe04c8e8-10
%fe04c8d8 00031c3f 00000000 00000000 ffc20010
%fe04c8e8 00000010 00000000 fe040001 ffc20010
%fe04c8f8 00000010 00000000 fe040001 ffe900a8
%fe04c908 fe0c767c fe0c0ee0 00000000 00000000
%fe04c918 00000000 00000000 00000000 00000000
%fe04c928 00000000 00000000 00000000 00000000
%fe04c938 00000000 00000000 00000000 00000000
%fe04c948 00000000 00000000 00000000 00000000

# .mo ffc2
ffc2 semstruc

# .d sem32 %fe04c8e8

Type: Private Event
Flags: Reset
pMuxQ: 00000000
Post Count: 0000
Open Count: 0001
Create Addr: 0010fe04
```

```
# .P#
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
0047# 000d 000a 000d 0004 blk 0200 7bd1f000 7bdfa188 7bde06b8      11 PMSHL32
```

In this example we are interested in slot 47. Its BlockId is owned by **vmkrhrw**

We dump the heap block from 0x10 bytes before the start.

Note that the low order bit of the header is 0, therefore a regular block.

Since the two low order bit are flags and the size is in double words we conveniently ignore these to obtain the size in bytes, which happens to be 0x10.

The block is owned by ffc2, which .mo tells us is **semstruc**.

This is very good news because all the semstruc owner relates to the 32-bit semaphore APIs. The .D command formats these for us.

Finally note that if we attempt to look at this from the application perspective we see from .P that the TSD is swapped out (Disp is blank). This means that the user registers for slot 47 can't be loaded. Furthermore attempts to look at the registers are unpredictable as DF and KDB will have not changed the values since they last loaded registers.

This is a case where BlockId analysis will give us a clue even if the application data is unavailable.

#### Lastly we look at an extended heap block.

```
# .s 4b
Current slot number: 004b

# .pb #
Slot  Sta BlockID  Name      Type      Addr      Symbol
004b# blk 21a0ade0 WKSTAHLp

# .m 21a0:ade0

*har      par      cpq      va      flg next prev link hash hob      hal
0003 %fef1f04c 00001000 %fdf1f000 001 0002 0020 0000 0000 0003 0000      =0000
hob      har hobnxt flgs own  hnte  sown,cnt lt st xf
0003 0003 ff08 0000 ffec 0000 0000 00 06 00 00 vmkrhrw

# dg 21a0
21a0 Code      Bas=fe070000 Lim=0000bd5b DPL=0 P RE      A

# dd %fe070000-10
%fe06ffff0 00000000 00000000 fe06fd72 4000bd6d
%fe070000 10d0006b 9090cbcb 9090cb90 000af390
%fe070010 3c600104 6aec8b55 8d026a01 16ebd866
%fe070020 6aec8b55 8d026a02 0aebd866 6aec8b55
%fe070030 8d026a00 46c6d866 561e00ee 21906857
%fe070040 1e7ec41f f714568b 750001c2 568b520e
%fe070050 27e2830e 29558826 2605eb5a 002945c6
%fe070060 4000c2f7 c0330574 f70d39e9 748000c2

# dd %fe070000-4+bd6c-10
%fe07bd58 fee2e9de 00000000 ff4c21a0 fdf1ff32
%fe07bd68 ffe9008c fe06fd70 fe02aa70 00000000
%fe07bd78 00000000 00000000 00000000 00000000
%fe07bd88 00000000 00000000 00000000 00000000
%fe07bd98 00000000 00000000 00000000 00000000
%fe07bda8 fe07bd6a ffe90048 00000201 00000000
%fe07bdb8 00000000 00000000 00000000 02010000
%fe07bdc8 00000000 00000000 00000000 000000ed
#

# .mo ff4c
ff4c fsd5

# .lml
hnte=0982 pmte=%fe0e1a14 mflags=0408b186 e:\ibmlan\netlib\splla.dll
hnte=097e pmte=%fe0e1a54 mflags=0408b186 e:\ibmlan\netlib\lrhml.dll
hnte=0979 pmte=%fe0e1bac mflags=0408b186 e:\ibmlan\netlib\lrns1.dll
hnte=096b pmte=%fe0e1d60 mflags=0408b186 e:\ibmlan\netlib\netibm.dll
hnte=0164 pmte=%fe02cc40 mflags=0498b1c8 e:\os2\dll\sysmono.fon
.
.
```

```

.
.
hmte=0181 pmte=%fe02ccb0 mflags=4498b1d5 e:\os2\dll\pmatm.dll
hmte=031b pmte=%fe02af18 mflags=0428a1c9 e:\ibmlan\netprog\netwksta.200
hmte=0306 pmte=%fe059f90 mflags=0428a1c9 e:\netware\nwifs.ifs
hmte=0160 pmte=%fe01ff4c mflags=0428a1c9 d:\dataex2\iwsfsd2.ifs
hmte=0117 pmte=%fdf5df60 mflags=0428a1c9 e:\os2\cdfs.ifs
hmte=00d2 pmte=%fdf53990 mflags=0428a1c9 e:\os2\hpfs.ifs

# .lmo 31b
hmte=031b pmte=%fe02af18 mflags=0428a1c9 e:\ibmlan\netprog\netwksta.200
seg  sect  psiz  vsiz  hob   sel   flags
0001 0003 2a8a ffdc 0000 2190 8d41 data prel rel
0002 0019 1d93 1d94 0000 2198 8d60 code shr prel rel
0003 0028 bd5c bd5c 0000 21a0 8d60 code shr prel rel
0004 0088 fd62 fd62 0000 21a8 8d60 code shr prel rel
0005 0108 606a 606a 0000 21b0 8d60 code shr prel rel
0006 0139 3492 3492 0000 21b8 8d60 code shr prel rel
0007 0154 002e 002f 0000 21c0 8c41 data prel
0008 0155 04bb 04bb 0000 21c8 8d60 code shr prel rel
#

```

What does BlockId 0x21a0ade0 represent?

We assume selector:offset and discover the owner is **vmkshrw**.

We dump the descriptor for selector 21a0 to find its base address.

Next we dump 0x10 bytes before the descriptor base to see the heap block header.

In this example the low order bit of the header is 1 so we have to look at the header extension for the owner information.

Adding the length to the base and backing off 0x10 bytes again we uncover the block header extension.

**Note:**

The following short cut could have been used:

```
dd %(21a0:0)-4+bd6c-10
```

In this case the owner is ff4c or **fsd5**. This is the 5th FSD to initialise.

We list the FSDs by using .LML and pick the 5th from the bottom. This turns out to be **netwksta.200**.

-----

## Blocking on a ChildWait

When a process calls **DosWaitChild** and blocks waiting for a child process to terminate, the BlockId is of the form:

```

ffcapppp      where pppp is the process id of the waiting
               thread.

```

The BlockId doesn't help us pin-point the processes being waited for.

All the child process have to be examined. The process status byte at offset 0xa into the Local information segment has either of the following bits set if the parent cares about termination of the child:

- 0x10                    The parent cares
- 0x20                    The parent did an exec-and-wait

The local information segment is embedded in the PTDA at the following offsets:

0x7ee	Retail 2.11
0x7f6	Debug 2.11
0x5be	Retail 3.0
05c6	Debug 3.0

-----

## Blocking on a RAMSEM

Potentially this is the most problematical type of wait to deal with. The BlockId is conventional and of the form:

```
fffexxxx    where xxxx is taken from the low order word
             of the user's RAMSEM.
```

There is no accountability associated with this type of semaphore. It is the responsibility of the user to manage their own accounting information. Accordingly most applications tend to imbed RAMSEMs into larger structures, which contain information such as use counts, owner identification, timeouts.

Two structures in particular are in common use:

[The Fast Safe RAMSEM](#)

[The PM Fast Safe RAMSEM](#)

The first step with RAMSEM BlockIds is to locate the user's RAMSEM address.

Next check ownership just in case this gives a clue to the associated process.

Ownership is indicated by a non-zero value in byte 0 of the RAMSEM. Very occasionally a RAMSEM is owned by the system. When this happens the ownership flags takes the value of the owning process.

We hope that the RAMSEM is embedded in either a Fast Safe RAMSEM or PM Fast Safe RAMSEM.

Both of these structures have a length prefix. The PM version is 0x12 and the non-PM version 0x0e.

Display storage before the RAMSEM and examine offset -0x12. Is this word 0x0012? If not then this is not a PM FSRS. Try -0xe. Does that contain 0x000e? If not then we will have to resort to more speculative analysis.

If either of these lengths correspond, look at the next two words, these contain the owning PID and TID. See whether this process and thread exists and what it is doing.

### Note:

Tid is sometimes 0 when there is only one thread in a process.

If this technique fails us then check the owner of the semaphore address, which is saved in **TCB\_SemInfo** and displayed by the **.PB** command. The owner of the semaphore, if it has not died, has to one of its accessors. If the RAMSEM is located in a Private Arena, then the owner is limited to one of the threads of the process that has blocked. If it is in shared storage, then the owned will be a thread in one of the processes on the VMCO chain. If we are lucky, the number of possibilities will be small, though this is not guaranteed.

The following example illustrates this technique:

```
>>> Slot 31 is blocked. Why?
```

```
.pb 31
Slot Sta BlockID Name      Type      Addr      Symbol
0031 blk fffe01ba aires    RamSem    e69f:000a
```

>>> Bad news! a RamSem. First check to see if its imbedded in a  
>>> FastSafeRamSem. We look at the RamSem address, back a few bytes

##.s 31

##dw e69f:000a-10  
Past end of segment: e69f:ffffffa

>>> It can't be a PM FSRamSem

##dw e69f:000a-a  
e69f:00000000 000e 0019 0000 0000 0000 01ff 01ba 0000  
e69f:00000010 0000 0000 0000 0000 0000 0000 0000 0000  
e69f:00000020 0000 0000 0000 0000 0000 0000 0000 0000  
e69f:00000030 0000 0000 0000 0000 0000 0000 0000 0000  
e69f:00000040 0000 0000 0000 0000 0000 0000 0000 0000  
e69f:00000050 0000 0000 0000 0000 0000 0000 0000 0000  
e69f:00000060 0000 0000 0000 0000 0000 0000 0000 0000  
e69f:00000070 0000 0000 0000 0000 0000 0000 0000 0000

>>> But it does look like a normal Fast Safe RamSem  
>>> Pid 19, Tid=0 (this is OK if just one thread in process 19).

##.p

Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
0001	0001	0000	0000	0001	blk	0100	ffe3a000	ffe3c7d4	ffe3c620	1e7c	00	*ager
0002	0001	0000	0000	0002	blk	0200	7b92a000	ffe3c7d4	7bb28020	1f3c	00	*tsd
0003	0001	0000	0000	0003	blk	0200	7b92c000	ffe3c7d4	7bb281d4	1f50	00	*ctxh
0004	0001	0000	0000	0004	blk	081f	7b92e000	ffe3c7d4	7bb28388	1f48	00	*kdb
0005	0001	0000	0000	0005	blk	0800	7b930000	ffe3c7d4	7bb2853c	1f20	00	*lazyw
0006	0001	0000	0000	0006	blk	0800	7b932000	ffe3c7d4	7bb286f0	1f3c	00	*asynchr
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmsshell
000d	0006	0001	0006	0002	blk	0800	7b940000	7bb460d0	7bb292dc	1ed4	01	pmsshell
000e	0006	0001	0006	0003	blk	0800	7b942000	7bb460d0	7bb29490		01	pmsshell
000f	0006	0001	0006	0004	blk	0800	7b944000	7bb460d0	7bb29644		01	pmsshell
0010	0006	0001	0006	0005	blk	0800	7b946000	7bb460d0	7bb297f8		01	pmsshell
0007	0006	0001	0006	0006	blk	0200	7b934000	7bb460d0	7bb288a4	1ecc	01	pmsshell
0013	0006	0001	0006	0007	blk	0200	7b94c000	7bb460d0	7bb29d14	1ecc	01	pmsshell
0015	0006	0001	0006	0008	blk	0200	7b950000	7bb460d0	7bb2a07c		01	pmsshell
0016	0006	0001	0006	0009	blk	0200	7b952000	7bb460d0	7bb2a230		01	pmsshell
0017	0006	0001	0006	000a	blk	0800	7b954000	7bb460d0	7bb2a3e4		01	pmsshell
0018	0006	0001	0006	000b	blk	0800	7b956000	7bb460d0	7bb2a598		01	pmsshell
0019	0006	0001	0006	000c	blk	0800	7b958000	7bb460d0	7bb2a74c	1eb8	01	pmsshell
001a	0006	0001	0006	000d	blk	0804	7b95a000	7bb460d0	7bb2a900	1ea8	01	pmsshell
001b	0006	0001	0006	000e	blk	0804	7b95c000	7bb460d0	7bb2aab4	1eb0	01	pmsshell
001c	0006	0001	0006	000f	blk	0500	7b95e000	7bb460d0	7bb2ac68	1ea8	01	pmsshell
001d	0006	0001	0006	0010	blk	0800	7b960000	7bb460d0	7bb2ae1c	1bb0	01	pmsshell
001e	0006	0001	0006	0011	blk	0800	7b962000	7bb460d0	7bb2afd0	1b8c	01	pmsshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
001f	0006	0001	0006	0012	blk	0200	7b964000	7bb460d0	7bb2b184	1eb8	01	pmsshell
0009	0007	0006	0007	0001	blk	0800	7b938000	7bb44020	7bb28c0c		00	harderr
0011	0007	0006	0007	0002	blk	0800	7b948000	7bb44020	7bb299ac		00	harderr
0012	0007	0006	0007	0003	blk	0800	7b94a000	7bb44020	7bb29b60		00	harderr
000a	0003	0000	0003	0001	blk	0200	7b93a000	7bb4484c	7bb28dc0		00	lanmsgex
000b	0004	0000	0004	0001	blk	080b	7b93c000	7bb45078	7bb28f74	1cf0	00	landll
000c	0005	0000	0005	0001	blk	0200	7b93e000	7bb458a4	7bb29128		00	lsdaemon
0014	0008	0000	0008	0001	blk	0200	7b94e000	7bb468fc	7bb29ec8		01	stoplan
0020	0009	0006	0009	0001	blk	0200	7b966000	7bb47128	7bb2b338		10	cmd
0021	000a	0006	000a	0001	blk	0500	7b968000	7bb47954	7bb2b4ec	1eb8	11	pmsshell
0023	000a	0006	000a	0002	blk	0200	7b96c000	7bb47954	7bb2b854	1ecc	11	pmsshell
0024	000a	0006	000a	0003	blk	0200	7b96e000	7bb47954	7bb2ba08	1eb8	11	pmsshell
0025	000a	0006	000a	0004	blk	0200	7b970000	7bb47954	7bb2bbbc		11	pmsshell
0026	000a	0006	000a	0005	blk	0200	7b972000	7bb47954	7bb2bd70	1ecc	11	pmsshell
0027	000a	0006	000a	0006	blk	0200	7b974000	7bb47954	7bb2bf24		11	pmsshell
0028	000a	0006	000a	0007	blk	0200	7b976000	7bb47954	7bb2c0d8		11	pmsshell
0029	000a	0006	000a	0008	blk	0200	7b978000	7bb47954	7bb2c28c		11	pmsshell
002a	000a	0006	000a	0009	blk	0200	7b97a000	7bb47954	7bb2c440		11	pmsshell
002c	000a	0006	000a	000b	blk	0200	7b97e000	7bb47954	7bb2c7a8	1eac	11	pmsshell
002d	000a	0006	000a	000c	blk	0200	7b980000	7bb47954	7bb2c95c	1eb8	11	pmsshell
002b	000d	0006	000d	0001	blk	0200	7b97c000	7bb48180	7bb2c5f4	1eb8	12	mrfilepm
0022	000d	0006	000d	0002	blk	0200	7b96a000	7bb48180	7bb2b6a0	1ecc	12	mrfilepm
002e	000f	000e	000f	0001	blk	0200	7b982000	7bb491d8	7bb2cb10		13	fvp
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
002f	000e	0006	000e	0001	blk	0200	7b984000	7bb489ac	7bb2ccc4		13	cmd
0030	0010	0006	0010	0001	blk	0200	7b986000	7bb49a04	7bb2ce78	1ed4	14	cmd
0031	0018	0010	0018	0001	blk	0200	7b988000	7bb4a230	7bb2d02c	1f00	14	aries
0032	0017	0006	0017	0001	blk	0400	7b98a000	7bb4aa5c	7bb2d1e0	1ed4	15	cmd
0033	0019	0017	0019	0001	blk	0300	7b98c000	7bb4b288	7bb2d394	1f00	15	orian

```
>>> Pid 19 is single threaded and is blocked. See what its Block-Id is.

##.pb 33
Slot Sta BlockID Name      Type      Addr      Symbol
0033 blk fffe01bb orian    RamSem    e66f:0000

>>> Once again a RamSem. This time there's no point in looking back
>>> a few bytes to see if it's imbedded in a FastSafeRamSem because
>>> the RamSem is allocated at the beginning of segment e66f.

>>> Our only hope is to see who else has access to this semaphore.

##.m e66f:0000

*har      par      cpq      va      flg next prev link hash hob      hal
0420 %fed03aca 00000010 %lccd0000 369 03f1 0075 0000 0000 051b 0000 hco=007ff
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
051b 0420 0000 4a2c ff82 04f1 0000 00 00 00 00 mshare
hco=007ff pco=fe85f816 hconext=007b6 hptda=04d1 f=16 pid=0019 a:orian.exe
hco=007b6 pco=fe85f6a9 hconext=00000 hptda=04fd f=17 pid=0018 a:aries.exe

>>> The RamSem is allocated in Named Shared Storage (mshare is the
>>> owner). The only two processes able to access this are Pids 19 and
>>> 18. Pid 19 is this thread, which we know doesn't own this RamSem
>>> since it's waiting for it. The leaves 18.

>>> We can't be certain from the evidence presented so far, but we can
>>> say:
>>> Either the RamSem is owned by 18, or it was owned by another
>>> thread that has since terminated. If it is owned by 18 then we
>>> have a deadlock between 18 and 19:
>>> orian.exe owns the FSRamSem and is waiting for the RamSem.
>>> aries.exe owns the RamSem and is waiting for the FSRamSem.
```

Fortunately simple RAMSEMs are becoming something of the past. And now that PM is 32-bit we will not see many Fast Safe RAMSEM either. We will look in detail later on at the semaphore structure that has replaced the FSRSEM in PM: the PMSEM and GRESEM.

## The MUX Wait

The last category of BlockIds to consider is the MUXWAIT. This has a BlockId of the form:

```
ffffdssss      where ssss is the slot id of the waiting thread.
```

A MUXWAIT is a multiplex semaphore wait. The semaphore comprising the MUX list may be:

RAMSEMs

SYSSEMs

32-bit Event & Mutex SEMs

We will consider each of these in turn.

The first step is to format the **mutexable**. This comprises 9-byte entries. +0x2 is the slot number of the waiter. +0x5 indicates the type of semaphore. +5 is the semaphore handle, which is interpreted according to type as follows:

```
0x00      Entry unused
0x01      handle is offset of SYSSEM from selector 400
0x02      Entry is a hob:offset of RAMSEM
```

0x03                   Entry is a physical address of a RAMSEM

0x04                   Entry points to a 32-bit Event SEM.

The following shows an example formatted MUXTABLE. There are up to 255 entries, but in practise the entries in use are grouped at the beginning of the table.

```
# db muxtable+(9*0) 19
0400:000048be c7 48 14 00 02 1a 07 be-00      GH.....>.
# db muxtable+(9*1) 19
0400:000048c7 d0 48 15 00 02 5c 07 be-00      PH...\>.
# db muxtable+(9*2) 19
0400:000048d0 ff ff 15 00 02 78 07 be-00      .....x.>.
# db muxtable+(9*3) 19
0400:000048d9 e2 48 1f 00 02 58 07 fa-03      bH...X.z.
# db muxtable+(9*4) 19
0400:000048e2 fd 48 1f 00 02 5c 07 fa-03      }H...\>.z.
# db muxtable+(9*5) 19
0400:000048eb c3 49 1f 00 02 50 07 fa-03      CI...P.z.
# db muxtable+(9*6) 19
0400:000048f4 57 49 58 00 02 61 01 64-07      WIX..a.d.
# db muxtable+(9*7) 19
0400:000048fd 06 49 1f 00 02 60 07 fa-03      .I...\>.z.
# db muxtable+(9*8) 19
0400:00004906 0f 49 1f 00 02 64 07 fa-03      .I...d.z.
# db muxtable+(9*9) 19
0400:0000490f 18 49 1f 00 02 68 07 fa-03      .I...h.z.
# db muxtable+(9*a) 19
0400:00004918 a8 49 1f 00 02 6c 07 fa-03      (I...l.z.
# db muxtable+(9*b) 19
0400:00004921 2a 49 58 00 01 f0 53 00-00      *IX..pS..
# db muxtable+(9*c) 19
0400:0000492a cc 49 30 00 03 02 a7 f1-00      LI0...'q.
# db muxtable+(9*d) 19
0400:00004933 3c 49 1b 00 01 9c 53 00-00      <I....S..
# db muxtable+(9*e) 19
0400:0000493c be 48 1b 00 03 da a6 f1-00      >H...Z&q.
# db muxtable+(9*f) 19
0400:00004945 4e 49 63 00 01 fc 53 00-00      NIc..|S..
# db muxtable+(9*10) 19
0400:0000494e eb 48 63 00 01 32 54 00-00      kHc..2T..
# db muxtable+(9*11) 19
0400:00004957 60 49 58 00 01 e4 53 00-00      `IX..dS..
# db muxtable+(9*12) 19
0400:00004960 ba 49 58 00 01 ea 53 00-00      :IX..jS..
# db muxtable+(9*13) 19
0400:00004969 72 49 57 00 01 ba 53 00-00      rIW...:S..
# db muxtable+(9*14) 19
0400:00004972 7b 49 57 00 01 c0 53 00-00      {IW...@S..
# db muxtable+(9*15) 19
0400:0000497b 84 49 57 00 01 c6 53 00-00      .IW...FS..
# db muxtable+(9*16) 19
0400:00004984 8d 49 57 00 01 cc 53 00-00      .IW...LS..
# db muxtable+(9*17) 19
0400:0000498d 96 49 57 00 01 d2 53 00-00      .IW...RS..
# db muxtable+(9*18) 19
0400:00004996 9f 49 57 00 01 d8 53 00-00      .IW...XS..
# db muxtable+(9*19) 19
0400:0000499f f4 48 57 00 01 de 53 00-00      tHW...^S..
# db muxtable+(9*1a) 19
0400:000049a8 b1 49 21 00 02 a8 07 fa-03      1I!...(z.
# db muxtable+(9*1b) 19
0400:000049b1 33 49 21 00 03 ee a6 f1-00      3I!...n&q.
# db muxtable+(9*1c) 19
0400:000049ba 45 49 58 00 01 f0 53 00-00      EIX..pS..
# db muxtable+(9*1d) 19
0400:000049c3 d9 48 1f 00 02 54 07 fa-03      YH...T.z.
# db muxtable+(9*1e) 19
0400:000049cc d5 49 00 00 00 00 00-00      UI.....
# db muxtable+(9*1f) 19
0400:000049d5 de 49 00 00 00 00 00 00-00      ^I.....
# db muxtable+(9*20) 19
0400:000049de e7 49 00 00 00 00 00 00-00      gI.....
# db muxtable+(9*21) 19
0400:000049e7 f0 49 00 00 00 00 00 00-00      pI.....
# dp %%fla6da 12
```

In this example there are only semaphore types 0, 1, 2 and 3. We will illustrate unravelling each of these in turn. For type 4 see the later section on 32-Bit Semaphores.

# The SYSSEM

## The SYSSEM

The SYSSEM block id points to a SYSSEM Table Entry.

### **Note:**

In a MUXWAIT only the offset is recorded in the MUX table entry. This should be used with selector 400.

In a single SYSSEM, the BlockId is the selector:offset to the SYSSEM Table Entry. The .PB command will display the SYSSEM name.

The example below is from a MUXWAIT which includes a SYSSEM

>> The MUXTABLE entry for slot 58. SYSSEM offset = 53f0

# db muxtable+(9\*b) 19

0400:00004921 2a 49 58 00 01 f0 53 00-00 \*IX..pS..

# .p 58

Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
0058	0014	0000	0014	0004	blk	021f	7bd30000	7bdfd260	7bde23f0	0eac	10	WKSTA

# .pb 58

Slot	Sta	BlockID	Name	Type	Addr	Symbol
0058	blk	ffff0058	WKSTA	MuxWait		

>> The SYSSEM Data Table Entry

>> slot = 0058

>> flag = 02

>> 01= waiting  
>> 02= mux waiting  
>> 04= owner (pid/tid) died  
>> 08= exclusive syssem  
>> 10= name entry needs removing  
>> 20= tid owner died  
>> 40= exit list thread owns this sem

>> reference count = 01

>> request count (by this owner) = 0

>> padding=00

# db 400:53f0 16

0400:000053f0 58 00 02 01 00 00 X.....

>> SYSSEM names are stored in a Record Management Package (RMP)

>> whose selector is the high word of:

# dd syssemrmphdl 11

0400:0000595a 04d00004

>> The RMP has a 0x14 byte header followed by variable length entries.

>> Each entry is prefixed with a word length followed by the entry data.

>> The entry data is the word offset of the corresponding SYSSEM Data Table

>> followed by offset 2 - n of the semaphore name. (the offset overlays

>> the first two bytes of the name which are always 'SE').

>>Scan the table looking for entry with offset 53f0...

# db 4d0:0

04d0:00000000 00 06 d0 02 0d 01 5b 03-01 00 00 00 00 04 00 00 ..P...[.....  
04d0:00000010 36 ff 00 00 10 00 5a 53-45 4d 5c 56 49 4f 50 4f 6.....ZSEM\VIOPO  
04d0:00000020 50 55 50 00 10 00 60 53-45 4d 5c 56 49 4f 50 52 PUP...`SEM\VIOPR  
04d0:00000030 54 53 43 00 12 00 66 53-45 4d 5c 44 41 54 41 45 TSC...fSEM\DATAE  
04d0:00000040 58 2e 45 52 52 00 12 00-6c 53 45 4d 5c 44 41 54 X.ERR...lSEM\DAT  
04d0:00000050 41 45 58 2e 4c 4f 47 00-14 00 72 53 45 4d 5c 49 AEX.LOG...rSEM\I



```

04d0:00000060 50 43 51 55 45 55 45 2e-53 45 4d 00 12 00 78 53 PCQUEUE.SEM...xS
04d0:00000070 45 4d 5c 50 4d 44 52 41-47 2e 53 45 4d 00 14 00 EM\PMDRAG.SEM...
# d
04d0:00000080 7e 53 45 4d 5c 4c 4b 4e-45 44 30 30 30 2e 53 45 ~SEM\LKNED000.SE
04d0:00000090 4d 00 14 00 84 53 45 4d-5c 4c 4b 4e 45 44 30 30 M....SEM\LKNED00
04d0:000000a0 31 2e 53 45 4d 00 14 00-8a 53 45 4d 5c 4c 4b 4e 1.SEM....SEM\LKN
04d0:000000b0 45 44 30 30 32 2e 53 45-4d 00 14 00 90 53 45 4d ED002.SEM....SEM
04d0:000000c0 5c 4c 4b 4e 45 44 30 30-33 2e 53 45 4d 00 13 00 \LKNED003.SEM...
04d0:000000d0 96 53 45 4d 5c 53 4d 47-43 4f 4e 54 2e 53 45 4d .SEM\SMGCONT.SEM
04d0:000000e0 00 13 00 9c 53 45 4d 5c-50 4d 48 44 45 52 52 2e ....SEM\PMHDERR.
04d0:000000f0 53 45 4d 00 19 00 a2 53-45 4d 5c 4e 50 49 50 45 SEM..."SEM\NPIPE
# d
04d0:00000100 53 5c 52 49 50 56 41 4e-2e 57 4e 4b 00 19 80 00 S\RIPVAN.WNK....
04d0:00000110 00 79 02 5c 49 42 4d 4c-41 4e 5c 53 49 4e 47 4c .y.\IBMLAN\SINGL
04d0:00000120 45 2e 52 43 46 00 19 00-ae 53 45 4d 5c 54 49 4d E.RCF....SEM\TIM
04d0:00000130 45 58 45 43 5c 49 53 5c-4c 4f 41 44 45 44 00 17 EXEC\IS\LOADED..
04d0:00000140 00 b4 53 45 4d 5c 4d 41-47 4e 55 4d 5c 4d 41 49 .4SEM\MAGNUM\MAI
04d0:00000150 4e 2e 53 45 4d 00 16 00-ba 53 45 4d 5c 4e 45 54 N.SEM...:SEM\NET
04d0:00000160 5c 42 52 4f 57 53 4e 43-42 2e 30 00 16 00 c0 53 \BROWSNCB.0...@S
04d0:00000170 45 4d 5c 4e 45 54 5c 42-52 4f 57 53 4e 43 42 2e EM\NET\BROWSNCB.
# d
04d0:00000180 31 00 16 00 c6 53 45 4d-5c 4e 45 54 5c 42 52 4f 1...FSEM\NET\BRO
04d0:00000190 57 53 4e 43 42 2e 32 00-16 00 cc 53 45 4d 5c 4e WSNCB.2...LSEM\N
04d0:000001a0 45 54 5c 42 52 4f 57 53-4e 43 42 2e 33 00 16 00 ET\BROWSNCB.3...
04d0:000001b0 d2 53 45 4d 5c 4e 45 54-5c 42 52 4f 57 53 4e 43 RSEM\NET\BROWSN
04d0:000001c0 42 2e 34 00 16 00 d8 53-45 4d 5c 4e 45 54 5c 42 B.4...XSEM\NET\B
04d0:000001d0 52 4f 57 53 4e 43 42 2e-35 00 16 00 de 53 45 4d ROWSNCB.5...^SEM
04d0:000001e0 5c 4e 45 54 5c 42 52 4f-57 53 4e 43 42 2e 36 00 \NET\BROWSNCB.6.
04d0:000001f0 18 00 e4 53 45 4d 5c 4e-45 54 5c 48 4f 53 54 41 ..dSEM\NET\HOSTA
# d
04d0:00000200 4e 4e 43 2e 53 45 4d 00-1d 00 ea 53 45 4d 5c 4e NNC.SEM...jSEM\N
04d0:00000210 45 54 5c 57 4b 53 54 41-5c 49 4e 54 45 52 47 54 ET\WKSTA\INTERGT
04d0:00000220 2e 53 45 4d 00 1d 00 f0-53 45 4d 5c 4e 45 54 5c .SEM...pSEM\NET\
04d0:00000230 57 4b 53 54 41 5c 52 45-4c 4f 47 4f 4e 2e 53 45 WKSTA\RELOGON.SE
04d0:00000240 4d 00 0f 00 f6 53 45 4d-5c 4d 53 52 56 57 55 30 M...vSEM\MSRVWU0
04d0:00000250 00 14 00 a8 53 45 4d 5c-4c 4b 4e 45 44 30 30 34 ...(SEM\LKNED004
04d0:00000260 2e 53 45 4d 00 14 00 02-54 45 4d 5c 4c 4b 4e 45 .SEM....TEM\LKNE
04d0:00000270 44 30 30 35 2e 53 45 4d-00 12 80 0d 01 5b 03 4e D005.SEM.....[.N

```

```

>> We find the entry at 4d0:227
>> The semaphore name is SEM\NET\WKSTA\RELOGON.SEM

```

## The MUX RAMSEM

In a MUX wait the RAMSEM id is recorded as a hob:offset.

In this example we look at the RAMSEM being waited on by slot 1f.

```

>> The MUX table entry:
>> slot = 1f, type = 2, hob= 03fa, offset=0758
0400:000048d9 e2 48 1f 00 02 58 07 fa-03 bH...X.z.
# db muxtable+(9*4) 19

>> Use .MOC to find the linear address
# .moc 3fa
*har      par      cpq      va      flg next prev link hash hob      hal
039b %fef23f5c 00000010 %1a350000 379 0413 039c 0000 0000 03fa 0000 hco=00c45
hob      har hobnxt flgs own hmte sown,cnt lt st xf
03fa 039b 0000 082c 03fb 03fb 0000 00 00 00 00 shared e:pmsshapi.dll
hco=0c45 pco=ffe77d74 hconext=00ee0 hptda=0873 f=16 pid=00e0 e:cmd.exe

>> This is owned by PMSHAPI. LN gives a lable.
# ln %1a350758
%1a350750 PMSHAPI ASEMRS + 8
#

```

# MUX Physical RAMSEM

In this example the MUX wait entry is for a physical address of a RAMSEM. A physical address would be used where the RAMSEM is in instance data - that makes it unique among RAMSEMs providing the RAMSEM is not swappable.

In this example the waiting slot is 1b

```
>> Mux table entry for slot 1b, type=3 (physical RAMSEM)
>> The physical address of the RAMSEM is %%00f1a6da
>> We need to determine the owner of this address.

# db muxtable+(9*e) 19
0400:0000493c be 48 1b 00 03 da a6 f1-00                >H...Z&q.

>> Display the page frame structure for frame 00f1a:
# .mp f1a
ffe24538 InUse: pVP=ff4076ce RefCnt=0003 Flg=0 ll=01 sl=00 Blk=0006a Frame=00f1a

>> Now display the virtual page structure to see who has backed this
>> frame:
# .mv %ff4076ce
VPI=057b pVP=ff4076ce SOW Frame=0f1a Flg=9d0 HobPg=0000 Hob=03df Ref=011

>> Now we have the hob and page offset into the hob. Display the
>> linear address of the Hob using .MOC, add the page offset and
>> the byte index from the physical address to obtain the
>> virtual address of the RAMSEM

# .moc 3df

*har      par      cpq      va      flg next prev link hash hob      hal
0382 %fef23d36 00000010 %1a260000 379 0381 0383 0000 0000 03df 0000 hco=00f37
hob      har hobnxt flgs own hmtc sown,cnt lt st xf
03df 0382 0000 082c 03da 03da 0000 00 01 00 00 shared      e:pmwin.dll
hco=0f37 pco=ffe78c2e hconext=00e8b hptda=0873 f=16 pid=00e0 e:cmd.exe

>> RAMSEM is at %1a260000+00000000+6da
>> RAMSEM is owned by pmwin.dll

# ln %1a2606da
No Symbols Found

>> LN doesn't work so thunk to a selector:offset and try again
>> cheat by looking up the selector assigned to pmwin in its
>> segment table:
# .lmo 3da
hmtc=03da pmte=%fdf21c14 mflags=0498b194 e:\os2\dll\pmwin.dll
obj  vsize  vbase  flags  ipagemap cpagemap hob sel
0001 0000f6f8 1a1b0000 80005025 00000001 00000010 03e9 d0df r-x shr alias conf
0002 0000c24e 1a1c0000 80005025 00000011 0000000d 03e8 d0e7 r-x shr alias conf
0003 00008c84 1a1d0000 80005025 0000001e 00000009 03e7 d0ef r-x shr alias conf
0004 0000b6e2 1a1e0000 80005025 00000027 0000000c 03e6 d0f7 r-x shr alias conf
0005 0000eb10 1a1f0000 80005025 00000033 0000000f 03e5 d0ff r-x shr alias conf
0006 00006292 1a200000 8000d025 00000042 00000007 03e4 d106 r-x shr alias conf iopl
0007 00003738 1a210000 8000d025 00000049 00000004 03e3 d10e r-x shr alias conf iopl
0008 000010c5 1a220000 80009025 0000004d 00000002 03e2 d116 r-x shr alias iopl
0009 000124d4 1a230000 80003025 0000004f 00000013 03e1 d11f r-x shr alias big
000a 000070ca 1a250000 80001025 00000062 00000008 03e0 d12f r-x shr alias
000b 00000ada 1a260000 80001063 0000006a 00000001 03df d137 rw- shr prel alias
000c 00001478 1a270000 80003063 0000006b 00000002 03de d13f rw- shr prel alias big
000d 000023f8 1a280000 80001063 0000006d 00000003 03dd d147 rw- shr prel alias
000e 00006444 1a290000 80001063 00000070 00000002 03dc d14f rw- shr prel alias
000f 00000142 1a2a0000 80001063 00000072 00000001 03db d157 rw- shr prel alias
0010 00000018 1a2b0000 80002063 00000073 00000001 03d9 d15f rw- shr prel big
0011 000003b8 1a100000 80002079 00000074 00000001 051e b087 r-- rsrc disc shr prel big
0012 00000dcc 1a1c0000 80002069 00000075 00000001 0509 b0e7 r-- rsrc shr prel big
0013 0000ffbc 1a210000 80002029 00000076 00000010 0504 b10f r-- rsrc shr big
0014 000002f0 00000000 00002039 00000086 00000001 0000 0000 r-- rsrc disc shr big
0015 00003524 1a120000 80002029 00000087 00000004 051b b097 r-- rsrc shr big

# ln d137:6dad137:000006da PMWIN MSGQUEUESEM1
```

---

# Structured Semaphores

We have discussed the following type of semaphore:

RAMSEM

SYSSEM

FSRAMSEM

PMFSRAMSEM

There are three others that occur with regularity in the system:

KSEM

32-bit SEM

GRESEM/PMSEM

---

## The Kernel Semaphore (KSEM)

The kernel semaphore is a RAMSEM and EVENT SEM with accountability in-built.

Many system control block have imbedded KSEMs. Included among these are the PTDA and MFT.

Some KSEMs are allocated out of the kernel heaps and have the owner mnemonic **KSEM**.

When a thread blocks on a KSEM the address (or handle) of the KSEM, or a field within the KSEM, is used as the BlockId depending upon the KSEM type.

For MUTEX KSEMs                      The BlockId is the address of the beginning of the [KSEM structure](#).

For Shared KSEMs                      The BlockId is the address of the Pending Readers count field within the [KSEM structure](#).

For Exclusive KSEMs                   The BlockId is the address of the Pending Writers count field within the [KSEM structure](#).

Under the debug (ALLSTRICT) kernel the KSEM contains an additional signature 'KSEM'. Always check a BlockId address to see if the 'KSEM' signature is present.

[.D KSEM](#) will format the KSEM.

In this example we look at Slot 6c to find out why it will not run.

```
#.pb 6c
Slot  Sta BlockID Name      Type      Addr      Symbol
006c  blk 7bdfc910 DEMO1

# .m 7bdfc910
*har      par      cpg      va      flg next prev link hash hob   hal
0087 %fef1fba4 00000082 %7bdf5000 121 0085 0088 0000 0000 0089 0000 =0000
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0089 0087 0000 0325 ffc9 0000 0000 00 00 00 00 ptda

>> This thread is blocked on an address in (its) PTDA. All PTDA
>> semaphores are KSEMs.

# .d KSEM %7bdfc910
Signature      : KSEM                      Nest: 0001
Type           : MUTEX
Flags          : 00
Owner          : 0041                      PendingWriters: 0001
```

```
>> So the owner is Slot 41. Lets look at him to see what he's up to.

# .pb 41
Slot Sta BlockID Name      Type      Addr      Symbol
0041 blk 04085ca7 DEMO1
# .m #408:5ca7
*har      par      cpq      va      flg next prev link hash hob      hal
0079 %fef1fa70 00000010 %7bf27000 129 0078 0077 0000 0000 007b 0000      sel=0408
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
007b 0079 0000 0324 ffa1 0000 0000 00 00 00 00 sft
#

>> Slot 41 is blocked waiting for some file system activity to complete.
>> We looked at this slot some time ago and found out that it was
>> waiting to close a device driver.
```

## The 32-Bit Semaphore Event and Mutex Semaphores

Block ids for 32-bit sems point to kernel heap allocated structure with object mnemonic **semstruc**.

**.PB** under the KDB usually identifies these as **SEM32**, but DF doesn't.

**.D SEM32** will format a 32-bit semaphore structure.

There are several structures that relate to 32-bit semaphores. Each of these is allocated from the kernel heaps and is assigned the following meaningful owner id mnemonics:

semmuxq (0xffbe)	Semaphore Mux Queue. This records instances of single event or mutex semaphores being also waited on in a mux wait.
semopenq (0xffbf)	Semaphore Open Queue. This tracks all processes that have opened a 32-bit semaphore.
semrec (0xffc0)	SemRecord. This is a system copy of the user's SemRecord structure, which was created when a Mux wait was declared. It correlates user semaphore ids with semaphore handles.
semstr (0xffc1)	The semaphore name string.
semstruc (0xffc2)	The main 32-bit structure. The address of this forms the Blockid when a thread waits on a 32-bit semaphore.

Of the associated structures listed above the Open Queue and Mux Queue may be formatted using:

**.D OPENQ**

**.D MUXQ**

In this example we look at the Blockid slot 42 is waiting on.

```
# .p 42 Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0042 000d 000a 000d 0002 blk 0200 7bd1a000 7bdfa188 7bddfe20 0ed4 11 PMSHL32

# .pb42
Slot Sta BlockID Name      Type      Addr      Symbol
0042 blk fe0bf91c PMSHL32

>> check owner of blockid
# .m %fe0bf91c

*har      par      cpq      va      flg next prev link hash hob      hal
0003 %fef1f04c 00001000 %fdf1f000 001 0002 0020 0000 0000 0003 0000      =0000
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0003 0003 ff08 0000 ffec 0000 0000 00 06 00 00 vmkrhrw

>> kernel swappable heap. Check current user of heap block.

# dd %fe0bf91c-10
%fe0bf90c ffbf000c 00010008 fe0bff20 ffc20014
%fe0bf91c 00000011 00000000 fef1ef94 fcae5a28
```

```

%fe0bf92c ffbf0010 00010006 fe08f24c fe0bf92e
%fe0bf93c ffbf000c 00010006 fe08f7c0 ffa4000c
%fe0bf94c fe0567d0 00010494 ffbf000c 0001000a
%fe0bf95c 00000000 ffa4000c fe0bf970 000104ec
%fe0bf96c ffa4000c fef1ef4c 000200ba ffbf000c
%fe0bf97c 00010005 fe0bf758 ffbf000c 00010005

# .mo ffc2
ffc2 semstruc

>> This is a 32-bit Semaphore

# .d sem32 %fe0bf91c
      Type: Shared Event
      Flags: Reset
      pMuxQ: 00000000
      Post Count: 0000
      pOpenQ: fef1ef94
      pName: fcae5a28
      Create Addr: ffbf0010

# .d openq %fef1ef94

      PID      Open Count
      -----
      000d      0001
      000a      0001

# da %fcae5a28
%fcae5a28 WORKPLAC\LAZYWRIT.SEM

>> For interest look for the owner of the OPENQ:

# .m %fef1ef94
*har      par      cpg      va      flg next prev link hash hob      hal
0003 %fef1f04c 00001000 %fdflf000 001 0002 0020 0000 0000 0003 0000      =0000
hob      har hobnxt flgs own hmte sown,cnt lt st xf
0003 0003 ff08 0000 ffec 0000 0000 00 06 00 00 vmkrhrw

# dd %fef1ef94 -10
%fef1ef84 00000000 00000000 fef10001 ffbf000c
%fef1ef94 0001000d fe0bf958 ffc20018 00000009
%fef1efa4 00000000 00000000 fef1ef58 fcb18478
%fef1efb4 ffbf000c 0001000d 00000000 ffc20018
%fef1efc4 00000009 00000000 00000000 fef1efb8
%fef1efd4 fcb18458 ff910024 00000007 00000000
%fef1efe4 00000000 00000000 00000000 00000000
%fef1eff4 00000000 00000000 ffea0004 fef2a128
# .mo ffbffffbf semopenq

>> For interest look for the owner of the pName:

# .m %fcae5a28

*har      par      cpg      va      flg next prev link hash hob      hal
0021 %fef1f2e0 00001400 %fca5f000 121 0020 0022 0000 0020 0022 0000      =0000
hob      har hobnxt flgs own hmte sown,cnt lt st xf
0022 0021 0000 0225 ffef 0000 0000 00 04 00 00 vmkshrw
# dd %fcae5a28-10
%fcae5a18 003f1d1a 0007004a 52000020 0000ffc1
%fcae5a28 4b524f57 43414c50 5a414c5c 49525759
%fcae5a38 45532e54 5412004d 520000c8 0000ff60
%fcae5a48 005d004a 1b131b12 1b151b14 1b171b16
%fcae5a58 1b191b18 1b1b1b1a 1b1d1b1c 1b1f1b1e
%fcae5a68 1b211b20 1b231b22 1b251b24 1b271b26
%fcae5a78 1b291b28 1b2b1b2a 1b2d1b2c 1b2f1b2e
%fcae5a88 1b311b30 1b331b32 1b351b34 1b371b36

# .mo ffc1
ffc1 semstr

```

-----

# PMSEM/GRESEM

32-bit PM (WARP) and Graphics Engine use a composite semaphore structure to serialise their resources.

This semaphore has the structure:

+0x0	7 byte Signature. 'PMSEM' for PMWIN and 'GRESEM' for PMGRE
+0x7	386 semaphore byte (PM uses the <b>bts</b> instruction on this under 386 processors otherwise it uses the 486 <b>cmpxchg</b> on the pid/tid)
+0x8	Owner pid (word)
+0xa	Owner tid (word)
+0xc	Owner nested use count (long)
+0x10	Number of waiters
+0x14	Number of times sem used (zero unless Debug version of PM)
+0x18	Handle for event semaphore
+0x1c	Address of caller (zero unless Debug version of PM)

PM uses a technique of polling this semaphore by waiting on the imbedded event semaphore handle for a limited time.

This technique has the advantage of speed combined with accountability but a thread waiting for a PMSEM or GRESEM may appear blocked, ready or running depending on the polling cycle. However it will be executing in a routine with a name such as **PMRequestMutexSem**. If the PMMERGE symbols are loaded this is readily detected.

The PM and GRE SEMs are contiguous and located at label **pmSemaphores**.

The handle (linear address) of the PM/GRE Semaphore is passed on entry to PMRequestMutexSem and tends to be retained in the EDX register.

The following semaphores are defined by PM:

0	PMSEM ATOM
1	PMSEM USER
2	PMSEM VISLOCK
3	PMSEM DEBUG
4	PMSEM HOOK
5	PMSEM HEAP
6	PMSEM DLL
7	PMSEM THUNK
8	PMSEM XLCE
9	PMSEM UPDATE
10	PMSEM CLIP
11	PMSEM INPUT
12	PMSEM DESKTOP
13	PMSEM HANDLE
14	PMSEM ALARM
15	PMSEM STRRES
16	PMSEM TIMER
17	PMSEM CONTROLS

18	GRESEM GreInit
19	GRESEM AutoHeap
20	GRESEM PDEV
21	GRESEM LDEV
22	GRESEM CodePage
23	GRESEM HFont
24	GRESEM FontCntxt
25	GRESEM FntDvr
26	GRESEM ShMalloc
27	GRESEM GlobalData
28	GRESEM DbcsEnv
29	GRESEM SrvLock
30	GRESEM SelLock
31	GRESEM ProcLock
32	GRESEM DriverSem
33	GRESEM semIfiCache
34	GRESEM semFontTable

In this example one of the shell threads seems to be getting very little CPU, though is frequently ready:

```
# .p 3a
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
*003a# 000d 0005 000d 000a rdy  0200 abd61000 abe4b5b4 abe2ee60 0ee4 11 PMSHL32

# .r
eax=13e30025 ebx=00000000 ecx=000a000d edx=13e7b4d4 esi=ffffffff edi=0068e55c
eip=1bd0d7ea esp=00637f44 ebp=00637f60 iopl=2 -- -- -- nv up ei pl nz na po nc
cs=005b  ss=0053  ds=0053  es=0053  fs=150b  gs=0000   cr2=00000000   cr3=001ad000
005b:1bd0d7ea ff4a10          dec     dword ptr [edx+10]    ds:13e7b4e4=00000006

# ln
%1bd0d770 PMMERGE PMREQUESTMUTEXSEM + 7a

# db %edx
%13e7b4d4 50 4d 53 45 4d 00 00 00-10 00 01 00 02 00 00 00 PMSEM.....
%13e7b4e4 06 00 00 00 00 00 00 00-05 00 01 80 00 00 00 00 .....
%13e7b4f4 50 4d 53 45 4d 00 00 00-00 00 00 00 00 00 00 00 PMSEM.....
%13e7b504 00 00 00 00 00 00 00 00-06 00 01 80 00 00 00 00 .....
%13e7b514 50 4d 53 45 4d 00 00 00-00 00 00 00 00 00 00 00 PMSEM.....
%13e7b524 00 00 00 00 00 00 00 00-07 00 01 80 00 00 00 00 .....
%13e7b534 50 4d 53 45 4d 00 00 00-00 00 00 00 00 00 00 00 PMSEM.....
%13e7b544 00 00 00 00 00 00 00 00-08 00 01 80 00 00 00 00 .....

>> PMSEM owner is pid 10 tid 1 and it has been requested twice by
tid/pid 10/1. There are 6 waiting threads.

# .p 42
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
0042  0010 0005 0010 0001 blk  0500 abd69000 abe4c19c abe2fe20 0ed8 13 MRFILEPM

# .pb 42
Slot  Sta  BlockID  Name      Type      Addr      Symbol
0042  blk  fdf8841c MRFILEPM

>> The owner is blocked.

# .m %fdf8841c

*har      par      cpq      va      flg next prev link hash hob      hal
```

```

0003 %feeeef04c 00001000 %fdeef000 001 0002 0021 0000 0000 0003 0000      =0000
hob   har hobnxt flgs own  hmte  sown,cnt lt st xf
0003  0003 ff05  0000 ffec 0000  0000 00  02 00 00 vmkrhrw

# dd %fdf8841c-10 14
%fdf8840c  000101c1 00000000 fdf88406 ffc20018

# .mo ffc2
ffc2 semstruc

# .d sem32 %fdf8841c
      Type: Shared Event
      Flags: Reset
      pMuxQ: 00000000
Post Count: 0000
      pOpenQ: fde305f8
      pName: NULL (anonymous)
Create Addr: abe2fe20

# .d openq %fde305f8

PID    Open Count
-----
0010    0001

#
# ln pmsemaphores
9f3f:0000b4b4 PMMERGE PMSEMAPHORES
# dl 9f3f
9f3f Data      Bas=13e70000 Lim=0000ffff DPL=3 P  RW    A
#
>> The sem we were waiting on was at %13e7b4d4 so must be the
>> USER SEM.

```

## Involuntary Suspension

In this section we discuss the mechanisms involved when a thread involuntarily gives up CPU processing time. That is, another thread independently causes a thread not to receive or to give up its time-slice.

The mechanisms available that cause suspension are:

Pre-emption

Another thread of a high priority becomes ready.

The suspended thread becomes ready and the pre-empting thread runs.

### Note:

Pre-emption is not possible when running in kernel-mode (specifically when **InDos** is non-zero, which is set shortly after entry to the kernel). Within the kernel co-operative multi-tasking operates: threads must yield explicitly (call the scheduler) to give up the processor. This applies equally to device drivers and file system drivers, which also run in kernel mode. Physical Device Drivers may use **DevHlp\_Yield** and **DevHlp\_TCYield** to give up the processor to other threads.

Critical Section

Another thread in the same process enters critical section.

The critical section thread runs and none of the other threads will run except if a signal 'fires'. If another ready thread in the same process is selected by the dispatcher for running it is held on a temporary queue with its status set to **crt**.

### Note:

The Critical Section thread has **run** status.

DosSuspendThread

Another thread in the same process has issued DosSuspendThread.

The suspending thread runs and the suspended thread enters **frz** state.



## Freeze Process

Either a Session Manager switch is in progress, a new process has been created suspended, a Virtual Device Driver has called the VDHFreezeVDM helper routine or the DosDebug DBG\_C\_Freeze command has been executed against a debuggee process.

The frozen process has a state of **frz** in all its threads.

Voluntary suspension is indicated by the **blk** state.

When a thread is suspended involuntarily it will normally be in one of the following states:

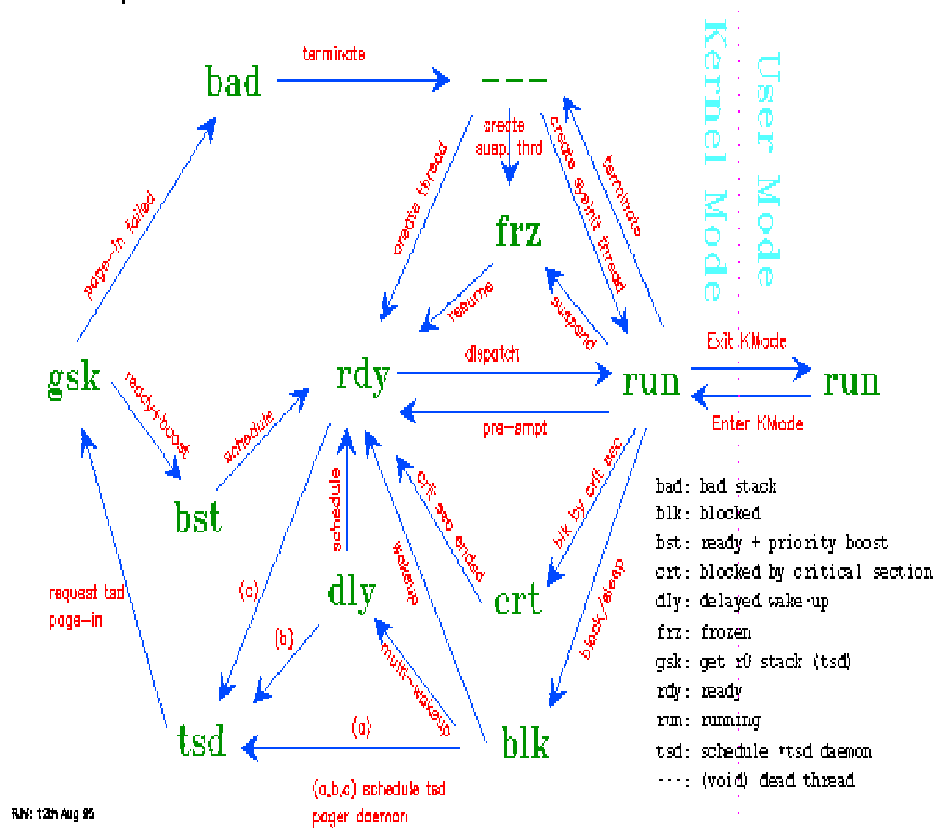
rdy	Ready and waiting to run.
crt	Ready but prohibited from dispatch by a critical section thread.
frz	Frozen or Suspended by freeze-process or DosSuspendThread.

The remaining six thread states related to transient system processing on behalf of a thread. These are:

dly	Delayed wake-up. Multiple threads have been woken from a blocked state because they were all waiting on the same BlockId and a multiple wake-up was specified to ProcRun. Each delayed thread is queued pending scheduling where priority recalculation and the thread's ring 0 stack is checked for presence in memory. If all is well then the thread is placed on the ready queue pending dispatch. If not, then the thread is placed on the TSD Daemon's queue for paging in the thread's TSD (ring 0 stack).
tsd	The thread is on the TSD Daemon's queue waiting for ring 0 stack page-in. The Daemon runs as an internal thread, which is labelled *tsd by the .P command. This thread is responsible for calling the page manager to page in a thread's TSD. Because a paging operation involves I/O and is therefore relatively slow, this operation is performed under the control of a separate thread. This allows other threads to be processed while the paging operation takes place.
gsk	Get Stack request in progress. The TSD Daemon is waiting for the Page Manager to signal completion of the paging I/O operation. Effectively a thread in this state is blocked waiting for completion of a TSD paging I/O request.
bst	Boosted Ready State. When the TSD page-in completes successfully, the thread is placed on the dispatchers ready queue with a priority boost. This condition is indicated by the boosted ready state. Strictly speaking this is not an independent state since no operation is required to take the thread from <b>bst</b> to <b>rdy</b> .
bad	TSD page-in request has failed. This is a serious and terminal condition, which is not expected to occur. It is possible that an I/O error has occurred during the TSD page-in request.
---	The null state occurs very fleetingly during thread creation and termination. It signifies that the thread's environment is incomplete.

The complete set of scheduler states for a finite state machine, which is illustrated in the following diagram.

## OS/2 Scheduler Finite State Machine



## Pre-emption and Priority Calculation

A thread is pre-empted when higher priority work becomes ready to process. Under normal circumstances the pre-empting thread will run then give up its time-slice and eventually the original thread will be re-scheduled.

It is possible for a thread not to be re-scheduled if a higher priority thread will not give up the processor. However, the OS/2 scheduler applies dynamic boost to priorities according to resource requirements and makes priority comparisons based on a calculated priority. The elements involved in the priority calculation are the following:

### TCBPriClass

The thread's priority class. There are four classes, which in order of priority are:

3	Time-critical
4	Foreground Server (or fixed high)
2	Regular
1	Idle

### TCBPriLevel

The priority delta which may range from 0x00 to 0x1f.

### TCBPriClassMod

The priority boosts which may be any of the combined values:

Server and Regular class threads may pre-empt each other depending on priority boosts and delta.

The key to looking at pre-emption problems is to look for other CPU bound threads of a higher priority. In particular time-critical threads.

.P displays the current calculated priority for each thread.

-----

## Critical Sections

When a thread enters critical section it effectively suspends all other threads in its process. There is an exception to this. If a signal is sent to the process and a signal handler is registered, then thread 1 will be dispatched to run the signal handler regardless of critical section.

The critical section thread may voluntarily block.

Other threads may attempt to become ready. If this happens the dispatcher will temporarily suspend them in **crt** state.

The appearance of the **crt** state certainly guarantees that another thread in the same process is in critical section. However, the converse is not true: the absence of **crt** does not preclude another thread from being in a critical section.

If a thread running in Critical Section blocks on a resource owned by any other thread in the same process then a deadlock will result. Because of this it is unwise to call any System API when in Critical Section.

Thread running in Critical Section have their TCB address stored in their process's PTDA at ptda\_pTCBCritSec.

The following example illustrates locating the critical section thread in a process.

```
# .P
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
0001  0001  0000  0000  0001  blk  0100  ffe4b000  ffe4c7dc  ffe4c624  0e84  00  *ager
0002  0001  0000  0000  0002  blk  0200  7a49e000  ffe4c7dc  7b49c020  0f44  00  *tsd
0003  0001  0000  0000  0003  blk  0200  7a49f000  ffe4c7dc  7b49c1d8  0f54  00  *ctxh
0004  0001  0000  0000  0004  blk  0800  7a4a0000  ffe4c7dc  7b49c390  0f24  00  *kdb
0005  0001  0000  0000  0005  blk  0800  7a4a1000  ffe4c7dc  7b49c548  0f40  00  *lazyw
000a  0004  0000  0004  0001  blk  0200  7a4a6000  7b655068  7b49cde0  00  LANMSGEX
*000c# 0006  0000  0006  0001  blk  0804  7a4a8000  7b6560b0  7b49d150  0c94  00  CNTRL
000d  0006  0000  0006  0002  blk  0804  7a4a9000  7b6560b0  7b49d308  00  CNTRL
000e  0006  0000  0006  0003  blk  0804  7a4aa000  7b6560b0  7b49d4c0  0f04  00  CNTRL
000f  0006  0000  0006  0004  blk  0804  7a4ab000  7b6560b0  7b49d678  0f04  00  CNTRL
0010  0006  0000  0006  0005  blk  0804  7a4ac000  7b6560b0  7b49d830  00  CNTRL
0011  0006  0000  0006  0006  blk  0804  7a4ad000  7b6560b0  7b49d9e8  0cc4  00  CNTRL
0012  0006  0000  0006  0007  blk  0804  7a4ae000  7b6560b0  7b49dba0  0f04  00  CNTRL
0013  0006  0000  0006  0008  blk  0804  7a4af000  7b6560b0  7b49dd58  0cb0  00  CNTRL
0007  0007  0001  0007  0001  blk  0500  7a4a3000  7b6568d4  7b49c8b8  0ebc  01  PMSHL32
000b  0007  0001  0007  0002  blk  0800  7a4a7000  7b6568d4  7b49cf98  01  PMSHL32
0009  0007  0001  0007  0003  blk  0800  7a4a5000  7b6568d4  7b49cc28  01  PMSHL32
0014  0007  0001  0007  0004  blk  0800  7a4b0000  7b6568d4  7b49df10  01  PMSHL32
0015  0007  0001  0007  0005  blk  0800  7a4b1000  7b6568d4  7b49e0c8  01  PMSHL32
0006  0007  0001  0007  0006  blk  0200  7a4a2000  7b6568d4  7b49c700  01  PMSHL32
0018  0007  0001  0007  0007  blk  0200  7a4b4000  7b6568d4  7b49e5f0  0ed4  01  PMSHL32
0019  0007  0001  0007  0008  blk  0200  7a4b5000  7b6568d4  7b49e7a8  01  PMSHL32
001a  0007  0001  0007  0009  blk  0200  7a4b6000  7b6568d4  7b49e960  01  PMSHL32
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
001b  0007  0001  0007  000a  blk  0800  7a4b7000  7b6568d4  7b49eb18  01  PMSHL32
001c  0007  0001  0007  000b  blk  0800  7a4b8000  7b6568d4  7b49ecd0  01  PMSHL32
001d  0007  0001  0007  000c  blk  0800  7a4b9000  7b6568d4  7b49ee88  01  PMSHL32
001e  0007  0001  0007  000d  blk  0804  7a4ba000  7b6568d4  7b49f040  01  PMSHL32
001f  0007  0001  0007  000e  blk  0804  7a4bb000  7b6568d4  7b49f1f8  01  PMSHL32
0020  0007  0001  0007  000f  blk  0500  7a4bc000  7b6568d4  7b49f3b0  01  PMSHL32
0021  0007  0001  0007  0010  blk  0200  7a4bd000  7b6568d4  7b49f568  0ebc  01  PMSHL32
002f  0012  0007  0012  0001  blk  0200  7a4cb000  7b658140  7b4a0d78  15  CMD
002e  0011  0007  0011  0001  blk  0200  7a4ca000  7b65791c  7b4a0bc0  14  CMD
0026  000b  0007  000b  0001  blk  0400  7a4c2000  7b6570f8  7b49fe00  0ebc  12  CMD
0023  000a  0007  000a  0001  blk  0500  7a4bf000  7b654844  7b49f8d8  0ebc  11  PMSHL32
0024  000a  0007  000a  0002  blk  0200  7a4c0000  7b654844  7b49fa90  11  PMSHL32
0025  000a  0007  000a  0003  blk  0200  7a4c1000  7b654844  7b49fc48  0ebc  11  PMSHL32
0022  000a  0007  000a  0004  blk  0200  7a4be000  7b654844  7b49f720  11  PMSHL32
0027  000a  0007  000a  0005  blk  0200  7a4c3000  7b654844  7b49ffb8  0ed4  11  PMSHL32
0028  000a  0007  000a  0006  blk  0200  7a4c4000  7b654844  7b4a0170  11  PMSHL32
0029  000a  0007  000a  0007  blk  0200  7a4c5000  7b654844  7b4a0328  11  PMSHL32
002a  000a  0007  000a  0008  blk  0200  7a4c6000  7b654844  7b4a04e0  11  PMSHL32
002c  000a  0007  000a  000a  blk  0200  7a4c8000  7b654844  7b4a0850  0eb0  11  PMSHL32
002d  000a  0007  000a  000b  blk  0200  7a4c9000  7b654844  7b4a0a08  0ebc  11  PMSHL32
```

```

0008 0008 0007 0008 0001 blk 0800 7a4a4000 7b654020 7b49ca70 00 HARDERR
0016 0008 0007 0008 0002 blk 0800 7a4b2000 7b654020 7b49e280 00 HARDERR
0017 0008 0007 0008 0003 blk 0800 7a4b3000 7b654020 7b49e438 00 HARDERR
002b 0013 0011 0013 0001 blk 0200 7a4c7000 7b65588c 7b4a0698 14 DEMORUN
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0030 0014 0013 0014 0001 blk 0200 7a4cc000 7b658964 7b4a0f30 14 CMD
0031 0015 0014 0015 0001 crt 0809 7a4cd000 7b659188 7b4a10e8 0f24 14 DEMOCRT
0032 0015 0014 0015 0002 crt 080b 7a4ce000 7b659188 7b4a12a0 0f0c 14 DEMOCRT
0033 0015 0014 0015 0003 blk 080b 7a4cf000 7b659188 7b4a1458 0cc4 14 DEMOCRT
0034 0015 0014 0015 0004 crt 011e 7a4d0000 7b659188 7b4a1610 0f20 14 DEMOCRT
0035 0015 0014 0015 0005 crt 080f 7a4d1000 7b659188 7b4a17c8 0f0c 14 DEMOCRT
0036 0015 0014 0015 0006 blk 080a 7a4d2000 7b659188 7b4a1980 0c04 14 DEMOCRT
003c 0015 0014 0015 0007 blk 080a 7a4d8000 7b659188 7b4a23d0 0eb8 14 DEMOCRT
0038 0015 0014 0015 0008 blk 080a 7a4d4000 7b659188 7b4a1cf0 14 DEMOCRT
0039 0015 0014 0015 0009 blk 080a 7a4d5000 7b659188 7b4a1ea8 14 DEMOCRT
003a 0015 0014 0015 000a crt 080c 7a4d6000 7b659188 7b4a2060 0f0c 14 DEMOCRT
003b 0015 0014 0015 000b blk 080c 7a4d7000 7b659188 7b4a2218 0c80 14 DEMOCRT

```

```

# dd %7b659188+ptda_ptcbcritsec-ptda_start 11
%7b6596c0 7b4a23d0

```

```

# dw %7b4a23d0 12
%7b4a23d0 0007 003c

```

```

# .pb 3c
Slot Sta BlockID Name Type Addr Symbol
003c blk fffe0027 DEMOCRT RamSem 00bf:0024

```

In this example pid 15 is stuck, threads are either blocked or suspended by critical section.

From the PTDA we find the critical section TCB address. From this we can either scan the .P listing for the TCB address or look at the second word, which contains the slot number for the thread.

The critical section thread has blocked on a RAMSEM whose address is **00bf:0024**. Since the selector is less than **2007** this has to be in its private arena. This is significant: only another thread in the same process could possibly post this semaphore.

## Suspension and Freezing

Suspension is achieved by any thread in a process calling DosSuspendThread. There is no accounting information associated with this API. One must examine all threads in the process to see if they are functioning correctly.

Freezing occurs for a number of reasons:

- A new process has been created with the thread initially suspended.

- The Session Manager (Shell Process 1) has used DosSystemService to freeze all threads of a process while a screen group switch occurs.

- A Virtual Device Driver has called VDHFreezeVDM.

- A Debug thread has called DosDebug using the DBG\_C\_Freeze command.

Again, there is no accounting information kept for this state.

If a single thread exists in the frozen process, check the parent process's threads for correct functioning.

If all threads are frozen check the Shell Process 1 for correct processing.

## Priority Inversion

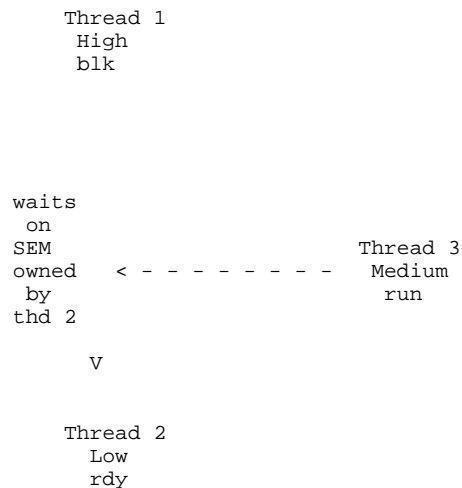
Priority Inversion is a hybrid situation that involves both the involuntary and voluntary suspension of two threads.

Consider the following:

A high priority thread is blocked on a resource.

A low priority thread owns the resource on which the high priority thread is blocked.

An independent thread of intermediate priority is running.



Thread 1 will not run until thread 2 gets a time-slice that allows it to run and release the semaphore thread 1 is waiting for.

Since thread 3 is a higher priority than thread 2 and is CPU bound, thread 2 never runs, nor does thread 1.

Thread 1's priority has effectively been reduced to that of thread 2's by a lower priority thread - thread 3. Thread 1 is said to have its priority inverted with respect to thread 3.

The Kernel implements an automatic inversion protection mechanism whenever a process blocks using a KSEM. Essentially this amounts to boosting the KSEM owner's thread priority by setting TCBPriorityMin to be just greater than the waiter's priority. This mechanism is implemented by the following three routines:

TKEnterInversion

Called to protect against priority inversion. For example, when a mutex KSEM is obtained increments TCBcBoostLock.

TKExitInversion

When an inversion protected KSEM is released TCBcboostlock is decremented. When TCBcBoostlock is zero and TCBPriorityMin and is not zero then it is set to zero and the priority recalculated.

TKDeclareInversion

Used to set the minimum priority of a thread to be waited on. If owner's TCBPriorityMin < waiter's TCBPriority and owner is in ready state then the owner's TCBPriorityMin to the waiter's TCBPriority+1.

For this mechanism to work, it must be possible to determine ownership from the semaphore so that TKDeclareInversion can determine which thread's priority to alter. It is also necessary to be able to determine whether raising the priority of thread will lead to other synchronisation problems or deadlocks through race conditions. Since the Kernel is a special case, and because pre-emption cannot occur while running in kernel mode, the kernel limits inversion protection to the KSEM only. Outside kernel mode, inversion is automatically protected against (for regular and foreground server threads) by application of the starvation priority boost.

-----

## Program Design Issues and Weaknesses

The following hit-list identifies potential weaknesses in program design that can lead to hang symptoms or serialisation problems:

1. Manipulation of thread priorities for the purpose of serialisation or sequencing execution is haphazard at best. At worst the performance of the entire system can be jeopardised.

The following guidelines should be applied when considering priority manipulation:

- Use priority delta to tell the system the relative importance of an application's threads.
  - Avoid priority class manipulation. Priority class tends to specify the relative importance of a thread with respect to all other threads in the system.
  - Avoid the use of time-critical priority. By setting this class, a thread is assuming the position of utmost importance in the system. This may not be a valid assumption for some system configurations and some users.
  - If priority class manipulation is desirable under some circumstances, then it should be parameterised so that it can be controlled as an option by the user.
2. If a window of exposure exists it will be exposed.
  3. Any common resource that is ever modified must have an associated lock or serialisation mechanism.
  4. Locks (serialisation techniques such as semaphores) that are concurrently held and waited on must be obtained in an established order.
  5. Simplistic approach (one lock) forces work to be channelled through a single-queue. Therefore design locks at the lowest level of contention.
  6. Distinguish process/data/repository serialisation otherwise an inconsistent system of locks may result:
    - Process locks are required where a only single instance of a process is allowed to operate. For example:
      - Finite State Machine state transitions;
      - Some FSM state users;
      - Any non-reentrant process.
    - Distinguishing Repository locks allows the repository is updated:
      - disk/directory reorganisation while file data is in use.
      - physical page assignments are allowed to change while data is in use - swapping
  7. Data optimisation: Artificial association of unrelated data items imposes serialisation constraints that will have two possible effects:
    - This necessitates unrelated processes to serialise.
    - Serialisation may lead to unavoidable deadlocks.
  8. Code optimisation: imposes process lock constraints in a similar way that data optimisation does.
  9. O-O tends to hide the data repository and structure. May even hide the process. Therefore designers need to consider whether locks are managed internally, within the object or explicitly. It may not be possible to handle the locks internally, because the context in which an instance method is being use (that is, the process) is not discernible from within the object.

-----

## Worked Examples

The following collection of worked examples illustrate how to use the debugging tools, in particular the Dump Formatter and Kernel Debugger to obtain information from a system under diagnosis.

The following topics are included:

### Finding File System information

This gives techniques for obtaining open file information and correlating open file names to handles and vice versa and finding out about record locking.

### [Exploring Memory Management](#)

This gives techniques for obtaining memory owner ownership and discovering who allocated a particular memory object. Name shared memory and Kernel heap memory are also discussed.

### [Exploring Presentation Manager](#)

This give techniques for finding Message Queues, Window Procedures and Window structures.

### [Debugging Ring 0 Loops from a Dump](#)

Dealing with ring 0 loops is relatively straight forward from the Kernel Debugger. It is also possible to diagnose Ring 0 problems from a dump if the current registers can be determined. This section given an example of using this technique in a File System Driver loop.

---

## How to Find File System Information

This section gives a basic overview of the file system control blocks, and shows how to answer the following questions:

1. What file system objects are open in a given process?
2. What file system objects are open in a VDM?
3. What file corresponds to a given handle?
4. What processes have opened a given file system object?

If the reader is unfamiliar with this subject then the sections that follow should be read in order:

These topics now follow:

[Finding files from handles](#)

[Finding files from handles in a VDM](#)

[Finding handles from file names](#)

[The Record Lock Record](#)

### **Note:**

The examples included in this section are worked on an OS/2 2.11 system. For OS/2 WARP the same techniques work, however the **SFTs**, whilst they may be located from the **SAS** in the same manner, they are allocated in segments that are mapped by different selectors. The effect of this is that short-cut techniques used to locate WARP **SFTs** may need to be re-worked.

---

## Finding Files From Handles

Open file system objects (files, named pipes, devices etc.) are represented by the **SFT** control block. The **SFT** contains three sections:

- Kernel data
- File system independent data
- File system dependent data

The kernel data section contains information to link the **SFT** to other system control blocks and to make the SFT usable by Kernel APIs. Of principle interest in this section are flags, handle, and pointer to the **MFT** and a chain pointer to other **SFTs** that represent other open instances of the same object. The kernel data is split into two discontinuous sections at each end of the **SFT**.

The file system independent data section contains information common to all **FSDs** needed to drive the file system. Of principle interest are the file attributes, open mode flags, opening process id and handle to the associated **VPB**.



The file system dependent data section is, as the name suggests, a work area private to the **FSD** that manages the file system object.

**Note:**

The **.D SFT command** formats the **SFT** always as if it is a **FAT** file. The information displayed in the file system dependent section may be misleading for non-FAT objects. The names of the fields formatted by **.D SFT** are prefixed **sfdFAT\_** for the file system dependent data so make it clear which information to treat with circumspection. The kernel and file system independent data name are prefixed with **sf\_** and **sfi\_** respectively.

When a file system object is opened, DosOpen returns a handle that represents the open object for all subsequent file system manipulation by the process until the object is closed. This handle is unique only within process and is referred to as the **JFN**. In protect mode processes the **JFN** is a 16-bit entity. In **VDMs**, however, to be consistent with **DOS** the **JFN** is an 8-bit entity, which may be correlated to the *real* **JFN** through a table in the **PDB**. This is illustrated later.

Each open file system object is also known by a system-wide unique handle, the **SFN**. Once the **SFN** is known then the corresponding **SFT** may be located and thence all file system information relating to the object.

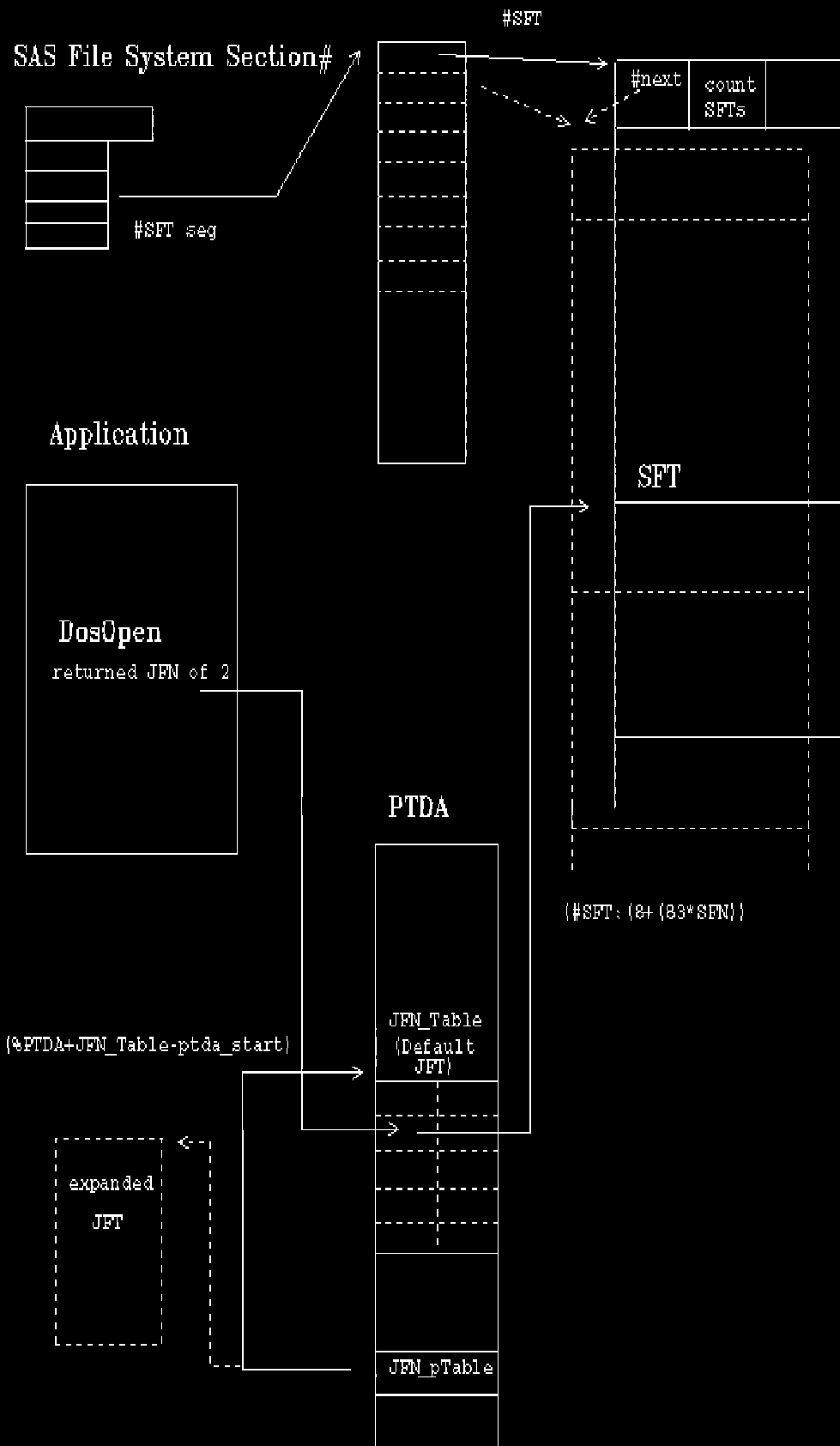
Each process is assigned by default a table of 20 words, which is indexed by the **JFN**. Each word of the **JFN\_table** contains the corresponding **SFN** for the open file. The default **JFN\_table** is imbedded within the **PTDA**. Prefixing the **JFN\_table** is a double-word pointer (**JFN\_ptable**) that points to this table. If the table is expanded (using **DosSetMaxFH**) then **JFN\_ptable** is updated to point to the current **JFN\_table**.

The key to finding information about open object in a given process is to locate **JFN\_ptable** and **JFN\_table**. Since both of these fields are part of the **PTDA** they may be referred to by name as symbols *for the current (system) context only*. For other contexts we may still use the **PTDA** symbols but in a relative fashion. The **PTDA** symbols are defined for the current process, which means that to use them successfully for another process, one must relocate them to the **PTDA** one wishes to reference. This is easily done by subtracting the label **PTDA\_START** from the desired symbol, then adding the address of the **PTDA** one wishes to see. For example: to see the **jfn\_table** field, enter

```
dw <ptda address>+jfn_table-ptda_start L2.
```

The relationships between the **JFN\_table**, **PTDA** and the **SFT** is illustrated in the following diagram:

# Open File - Application to System



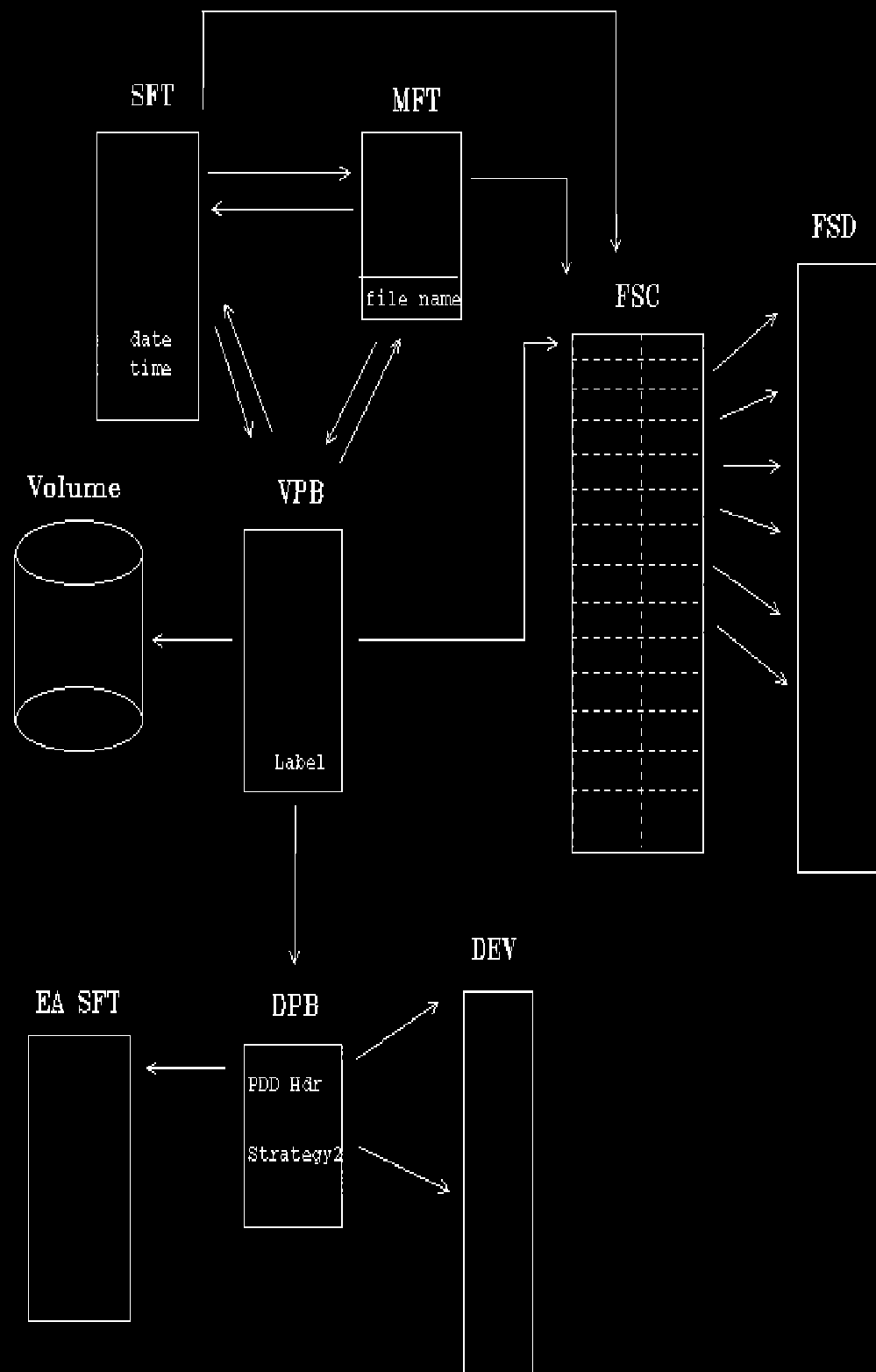
---

# File System Control Block Relationships

In the examples that follow, we explore the relationships between each of the major file-system control blocks. These relationships are illustrated in the following diagrams.

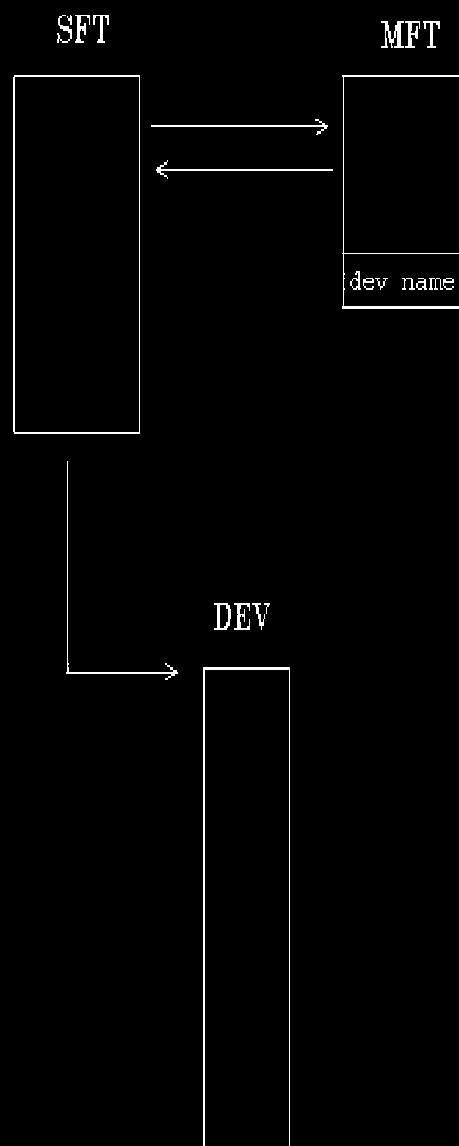
**Open File**

## Open File - System View



Open Device

## Open Device - System View



# Finding Files From Handles - Example

In the following example we choose to discover all the open file system objects in process 19, which happens to be running the IPFC compiler.

```
>>> List all the thread slots in the system to find IPFC
```

```
# .p
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
0001  0001  0000  0000  0001  blk  0100  ffe3a000  ffe3c7d4  ffe3c61c  1e7c  00  *ager
0002  0001  0000  0000  0002  blk  0200  7b7aa000  ffe3c7d4  7b9a8020  1f3c  00  *tsd
0003  0001  0000  0000  0003  blk  0200  7b7ac000  ffe3c7d4  7b9a81d8  1f50  00  *ctxh
0004  0001  0000  0000  0004  blk  081f  7b7ae000  ffe3c7d4  7b9a8390  1f48  00  *kdb
0005  0001  0000  0000  0005  blk  0800  7b7b0000  ffe3c7d4  7b9a8548  1f20  00  *lazyw
0006  0001  0000  0000  0006  blk  0800  7b7b2000  ffe3c7d4  7b9a8700  1f3c  00  *asynchr
0009  0002  0000  0002  0001  blk  021f  7b7b8000  7b9c4020  7b9a8c28  00  LOGDAEM
0008  0003  0001  0003  0001  rdy  061f  7b7b6000  7b9c484c  7b9a8a70  1eb8  01  PMSHL32
000b  0003  0001  0003  0002  blk  0800  7b7bc000  7b9c484c  7b9a8f98  01  PMSHL32
000c  0003  0001  0003  0003  blk  0800  7b7be000  7b9c484c  7b9a9150  01  PMSHL32
000d  0003  0001  0003  0004  blk  0800  7b7c0000  7b9c484c  7b9a9308  01  PMSHL32
000e  0003  0001  0003  0005  blk  0800  7b7c2000  7b9c484c  7b9a94c0  01  PMSHL32
0007  0003  0001  0003  0006  blk  0200  7b7b4000  7b9c484c  7b9a88b8  1ecc  01  PMSHL32
0011  0003  0001  0003  0007  blk  0200  7b7c8000  7b9c484c  7b9a99e8  1ecc  01  PMSHL32
0012  0003  0001  0003  0008  blk  0200  7b7ca000  7b9c484c  7b9a9ba0  01  PMSHL32
0013  0003  0001  0003  0009  blk  0200  7b7cc000  7b9c484c  7b9a9d58  01  PMSHL32
0014  0003  0001  0003  000a  blk  0800  7b7ce000  7b9c484c  7b9a9f10  01  PMSHL32
0015  0003  0001  0003  000b  blk  0800  7b7d0000  7b9c484c  7b9aa0c8  01  PMSHL32
0016  0003  0001  0003  000c  blk  0800  7b7d2000  7b9c484c  7b9aa280  01  PMSHL32
0017  0003  0001  0003  000d  blk  0804  7b7d4000  7b9c484c  7b9aa438  1ea8  01  PMSHL32
0018  0003  0001  0003  000e  rdy  0804  7b7d6000  7b9c484c  7b9aa5f0  01  PMSHL32
0019  0003  0001  0003  000f  blk  0500  7b7d8000  7b9c484c  7b9aa7a8  01  PMSHL32
001a  0003  0001  0003  0010  rdy  0801  7b7da000  7b9c484c  7b9aa960  1bac  01  PMSHL32
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
001b  0003  0001  0003  0011  blk  0800  7b7dc000  7b9c484c  7b9aab18  01  PMSHL32
*001c# 0003  0001  0003  0012  run  0800  7b7de000  7b9c484c  7b9aacd0  1b8c  01  PMSHL32
001d  0003  0001  0003  0013  blk  0200  7b7e0000  7b9c484c  7b9aae88  01  PMSHL32
0023  0018  0003  0018  0001  rdy  061f  7b7ec000  7b9c7128  7b9ab8d8  1eb8  13  EPM
0038  0018  0003  0018  0002  blk  0200  7b816000  7b9c7128  7b9adcf0  1ecc  13  EPM
0037  0013  0003  0013  0001  blk  0200  7b814000  7b9c9a04  7b9adb38  19  IBMAVSD
0033  0012  0003  0012  0001  blk  0200  7b80c000  7b9c89ac  7b9ad458  1eb8  17  PMDRAW
0035  0012  0003  0012  0002  blk  0200  7b810000  7b9c89ac  7b9ad7c8  1eb8  17  PMDRAW
0036  0012  0003  0012  0003  blk  0200  7b812000  7b9c89ac  7b9ad980  17  PMDRAW
0034  0010  0003  0010  0001  blk  0400  7b80e000  7b9c91d8  7b9ad610  1ed4  12  CMD
002e  000d  0003  000d  0001  blk  0200  7b802000  7b9c8180  7b9acbc0  1eb8  16  PULSE
0030  000d  0003  000d  0002  rdy  0100  7b806000  7b9c8180  7b9acf30  1f28  16  PULSE
002f  000d  0003  000d  0003  rdy  081f  7b804000  7b9c8180  7b9acd78  1f00  16  PULSE
002d  000c  0003  000c  0001  blk  0200  7b800000  7b9c7954  7b9aca08  1eb8  15  DINFO
0032  000c  0003  000c  0002  rdy  061f  7b80a000  7b9c7954  7b9ad2a0  1f00  15  DINFO
002c  000b  0003  000b  0001  blk  0200  7b7fe000  7b9c58a4  7b9ac850  1eb8  14  MRFILE32
0031  000b  0003  000b  0002  blk  0200  7b808000  7b9c58a4  7b9ad0e8  1ecc  14  MRFILE32
0029  000a  0003  000a  0001  rdy  061f  7b7f8000  7b9c68fc  7b9ac328  1eb8  10  PMDIARY
001f  0006  0003  0006  0001  rdy  062f  7b7e4000  7b9c60d0  7b9ab1f8  1eb8  11  PMSHL32
0021  0006  0003  0006  0002  blk  0200  7b7e8000  7b9c60d0  7b9ab568  11  PMSHL32
0022  0006  0003  0006  0003  blk  0200  7b7ea000  7b9c60d0  7b9ab720  1eb8  11  PMSHL32
0020  0006  0003  0006  0004  blk  0200  7b7e6000  7b9c60d0  7b9ab3b0  11  PMSHL32
001e  0006  0003  0006  0005  blk  0200  7b7e2000  7b9c60d0  7b9ab040  1ecc  11  PMSHL32
0024  0006  0003  0006  0006  blk  0200  7b7ee000  7b9c60d0  7b9aba90  11  PMSHL32
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
0025  0006  0003  0006  0007  blk  0200  7b7f0000  7b9c60d0  7b9abc48  11  PMSHL32
0026  0006  0003  0006  0008  blk  0200  7b7f2000  7b9c60d0  7b9abe00  11  PMSHL32
0027  0006  0003  0006  0009  blk  0200  7b7f4000  7b9c60d0  7b9abfb8  11  PMSHL32
0028  0006  0003  0006  000a  blk  0200  7b7f6000  7b9c60d0  7b9ac170  11  PMSHL32
002a  0006  0003  0006  000c  blk  021f  7b7fa000  7b9c60d0  7b9ac4e0  1eac  11  PMSHL32
002b  0006  0003  0006  000d  blk  0200  7b7fc000  7b9c60d0  7b9ac698  1eb8  11  PMSHL32
000a  0004  0003  0004  0001  blk  0800  7b7ba000  7b9c5078  7b9a8de0  00  HARDERR
000f  0004  0003  0004  0002  blk  0800  7b7c4000  7b9c5078  7b9a9678  00  HARDERR
0010  0004  0003  0004  0003  blk  0800  7b7c6000  7b9c5078  7b9a9830  00  HARDERR
0039  0019  0010  0019  0001  rdy  061f  7b818000  7b9ca230  7b9adea8  1f0c  12  IPFC
```

```
>>> IPFC is running in slot 39, but this is not the current system
```

```

>>> slot so we have to refer to PTDA symbols relative to pPTDA
>>> First establish whether JFN_table has been expanded?

# dw %7b9ca230 + jfn_ptable-ptda_start l2
%7b9caa18 fd8a 0030

>>> No it hasn't - it's still based on selector 30 and therefore
>>> still imbedded in the PTDA at label JFN_table.
>>> Note: we can't display it as 30:fd8a since selector 30
>>> aliases the current system PTDA, hence:

# dw %7b9ca230 + jfn_table-ptda_start l14
%7b9ca7e6 0027 0027 0027 0074 002a 0072 0077 0068
%7b9ca7f6 0015 0041 0069 007f ffff ffff ffff ffff
%7b9ca806 ffff ffff ffff ffff

>>> These are the SFNs that correspond to JFNs 0000 through 0014.
>>> In fact the highest JFN currently open in this process is 000b
>>> which corresponds to SFN 007f

>>> Next we locate the STF. From the SAS we look for the SFT selector:

# .a
--- SAS Base Section ---
        SAS signature: SAS
        offset to tables section: 0016
        FLAT selector for kernel data: 0168
        offset to configuration section: 001E
        offset to device driver section: 0020
        offset to Virtual Memory section: 002C
        offset to Tasking section: 005C
        offset to RAS section: 006E
        offset to File System section: 0074
        offset to infoseg section: 0080
--- SAS Protected Modes Tables Section ---
        selector for GDT: 0008
        selector for LDT: 0000
        selector for IDT: 0018
        selector for GDTPOOL: 0100
--- SAS Device Driver Section ---
        offset for the first bimodal dd: 0CB9
        offset for the first real mode dd: 0000
        sel for Drive Parameter Block: 04C8
        sel for ABIOS prot. mode CDA: 0000
        seg for ABIOS real mode CDA: 0000
        selector for FSC: 00C8
--- SAS Task Section ---
        selector for current PTDA: 0030
        FLAT offset for process tree head: FFF10910
        FLAT address for TCB address array: FFF06BB6
        offset for current TCB number: FFDFFB5E
        offset for ThreadCount: FFDFFB62
--- SAS File System Section ---
        handle to MFT PTree: FE72CFBC
        selector for System File Table: 00C0
        sel. for Volume Parameter Bloc: 0788
        sel. for Current Directory Struc: 07B8
        selector for buffer segment: 00A8
--- SAS Information Segment Section ---
        selector for global info seg: 0428
        address of curtask local infoseg: 03C80000
        address of DOS task's infoseg: FFFFFFFF
        selector for Codepage Data: 07CB
--- SAS RAS Section ---
        selector for System Trace Data Area: 04B0
        segment for System Trace Data Area: 04B0
        offset for trace event mask: 0B28
--- SAS Configuration Section ---
        offset for Device Config. Table: 0D50
--- SAS Virtual Memory Mgt. Section ---
        Flat offset of arena records: FFF13304
        Flat offset of object records: FFF1331C
        Flat offset of context records: FFF1330C
        Flat offset of kernel mte records: FFF0A891
        Flat offset of linked mte list: FFF07934
        Flat offset of page frame table: FFF11A70
        Flat offset of page range table: FFF111EC
        Flat offset of swap frame array: FFF03BAC

```



```

        Flat offset of Idle Head: FFF10090
        Flat offset of Free Head: FFF10080
        Flat offset of Heap Array: FFF11B78
        Flat offset of all mte records: FFF12E04

```

```

>>> We see this is assigned to selector c0.
>>> This is not quite the SFT but a table of selectors that point to
>>> each extent of the SFT. Each extent holds up to 500 STF entries.
>>> All the SFN's we're interested in are less than 500 so occupy the
>>> first extent. Note: we could have obtained the SFT selector from:

```

```

# ln GDT_SFT
138:000000c0 os2krnl DOSGDTDATA:GDT_SFT
#

```

```

>>> List the table of extents:

```

```

# dw c0:0
00c0:00000000  0438 0000 0000 0000 0000 0000 0000 0000
00c0:00000010  0000 0000 0000 0000 0000 0000 0000 0000
00c0:00000020  0000 0000 0000 0000 0000 0000 0000 0000
00c0:00000030  0000 0000 0000 0000 0000 0000 0000 0000
00c0:00000040  0000 0000 0000 0000 0000 0000 0000 0000
00c0:00000050  0000 0000 0000 0000 0000 0000 0000 0000
00c0:00000060  0000 0000 0000 0000 0000 0000 0000 0000
00c0:00000070  0000 0000 0000 0000 0000 0000 0000 0000

```

```

>>> Now list the first extent:

```

```

# dw 438:0
0438:00000000  0000 0000 0000 0000 0001 0000 0000 0001
0438:00000010  0000 0000 0800 c800 0000 0000 0000 0000
0438:00000020  7000 860e c6fe 6821 0008 0000 0000 0000
0438:00000030  0000 0000 0000 0000 0000 0000 0000 0000
0438:00000040  0000 0000 0000 0000 0000 8a30 007a 0000
0438:00000050  0000 0000 0000 a000 0000 1200 0000 0000
0438:00000060  0000 0000 0000 0000 e700 0110 6000 00ee
0438:00000070  0000 0000 0000 0000 0000 0000 0000 0000

```

```

>>> There is an 8 byte header to each extent. It followed by one or
>>> more 131 (hex 83) byte SFT entries. The first word of the header
>>> contains the selector for the next extent. In this case there
>>> isn't one.

```

```

>>> To locate the SFT entry corresponding to SFN we use the formula
>>> 438:(8+(83*SFN))
>>> We can dump this out directly or by using the .D SFT command:
>>> Start by examining SFN 0077 (JFN 0006 for slot 39)

```

```

# .d sft 438:(8+(83*77))
      sf_ref_count: 0001                      sfi_mode: 20a2
      sf_usercnt: 0000                      sfi_hVPB: 0012
      reserved: 00                          sfi_ctime: 0000
      sf_flags(2): 0000:0000                sfi_cdate: 0000
      sf_devptr: #0000:0000                 sfi_atime: 0000
      sf_FSC: #00c8:0008                    sfi_adata: 0000
      sf_chain: #0000:0000                  sfi_mtime: 6000
      sf_MFT: fe87ebf0                      sfi_mdate: 1b0b
sfdFAT_firFILEclus: 57e4                    sfi_size: 00000000
sfdFAT_cluspos: 0ff8                        sfi_position: 00000000
sfdFAT_lstclus: 0038                        sfi_UID: 0000
sfdFAT_dirsec: 00002cad                     sfi_PID: 0019
sfdFAT_dirpos: 09                          sfi_PDB: 0000
      sfdFAT_name: FCLDLGP DLL              sfi_selfsfm: 0077
sfdFAT_EAHandle: 0000                      sfi_tstamp: 00
      sf_plock: 0000                        sfi_DOSattr: 20
      sf_NmPipeSfn: 0000
      sf_codepage: 0000

```

```

>>> Fully qualified file system names are maintained in the Master
>>> File Table entries. Lets check out the MFT for this SFT, which
>>> is pointed to by sf_MFT.
>>> Under the Kernel debugger we could use .D MFT to format an MFT
>>> entry. Under the dump formatter .D MFT does not work correctly:

```

```

# db %fe87ebf0
%fe87ebf0 4b 53 45 4d 01 02 00 00-00 00 00 00 00 00 00 00 KSEM.....
%fe87ec00 00 00 ed 3c 38 04 00 00-00 00 4b 53 45 4d 01 02 ..m<8.....KSEM..

```

```
%fe87ec10 00 00 00 00 00 00 00 00 00-00 00 00 00 1c 0e 00 00 .....
%fe87ec20 6d 46 12 00 43 3a 5c 24-30 30 30 30 24 00 87 fe mF..C:\$0000$.~
%fe87ec30 14 00 9e ff 29 00 0b 00-dc eb 87 fe 08 43 83 fe ....)\k.~.C.~
%fe87ec40 f0 eb 87 fe 48 00 9e ff-4b 53 45 4d 01 02 00 00 pk.~H...KSEM....
%fe87ec50 00 00 00 00 00 00 00 00 00-00 00 5e 3a 38 04 00 00 .....^:8...
%fe87ec60 00 00 4b 53 45 4d 01 02-00 00 00 00 00 00 00 00 ..KSEM.....
```

```
>>> The file name is at MFT+34 in the ALLSTRICT kernel and 2a in the
>>> RETAIL kernel. There are two imbedded KSEMs, which only only
>>> contain the signature KSEM in the ALLSTRICT kernel, also the MFT
>>> contains the signature mF at +30 in the ALLSTRICT kernel.
>>> The first KSEM used for serialising read/single write access to
>>> the file. The second KSEM is used for updating the cluster map.
```

```
>>> These KSEMs can be formatted using .d KSEM
```

```
# .d ksem %fe87ebf0
Signature      : KSEM                      Nest: 0000
Type           : SHARE                    Readers: 0000
Flags          : 01                      PendingReaders: 0000
Owner          : 0000                    PendingWriters: 0000
# .d ksem %fe87ebf0+1a
Signature      : KSEM                      Nest: 0000
Type           : SHARE                    Readers: 0000
Flags          : 01                      PendingReaders: 0000
Owner          : 0000                    PendingWriters: 0000
#
```

```
>>> In this case they are unowned.
```

```
>>> The file name in the MFT does not agree with the sfdFAT_name.
>>> We suspect that this is not a FAT file. This can be verified by
>>> examining the file system control block entry for the FSD that's
>>> managing this file. The FSC entry address appears in the SFT at
>>> sf_FSC. In this case it is 00c8:0008.
```

```
# dw c8:8
00c8:00000008 0b68 0840 0b6c 0840 0000 0828 01fc 0828
00c8:00000018 0010 0828 05b4 0820 0570 0828 0580 0828
00c8:00000028 0634 0828 0640 0828 0e3c 0828 1120 0828
00c8:00000038 0834 0828 090c 0828 09f8 0820 1130 0828
00c8:00000048 1f24 0828 1f6e 0828 2122 0828 16e4 0828
00c8:00000058 1b10 0828 1b38 0828 1bec 0828 1dc8 0828
00c8:00000068 0c60 0820 0d70 0820 1f14 0828 215c 0828
00c8:00000078 22a0 0828 2294 0828 111c 0820 25fc 0828
```

```
>>> Each FSC entry is a table of far16 pointers. The first points the
>>> The FSD attributes, the second to the name and the remainder are
>>> standard FSD entry points. (See OEMI IFS Documentation).
>>> the name of this FSD is....
```

```
# da 840:b6c
0840:00000b6c HPFS
```

```
>>> The word prefixing the file name in the MFT is the handle to the
>>> Volume Parameter Block (hVPB). This also appears in the SFT under
>>> sfi_hVPB. In this instance the hVPB is 0012.
>>> To format the VPB we need to obtain the selector for the VPB
>>> segment. N.B. this is not stored in the SAS under Volume Parameter
>>> Block. We have to locate this using:
```

```
# ln GDT_VPB
138:00000098 os2krnl DOSGDTDATA:GDT_VPB
#
```

```
>>> The hVPB is an offset into the VPB segment. Format a VPB
>>> using .D VPB
```

```
# .d vpb 98:12
      vpb_flink: 0000          vpdFAT_cluster_mask: 02
      vpb_blink: 008d          vpdFAT_cluster_shift: 00
      vpb_ref_count: 0057      vpdFAT_first_FAT: 0000
      vpb_search_count: 0004   vpdFAT_FAT_count: 00
      vpb_first_access: 00     vpdFAT_root_entries: 0030
      vpb_signature: 444a      vpdFAT_first_sector: 06001100
      vpb_flags(2): 02:00      vpdFAT_max_cluster: 7d5c
      vpb_FSC: #00c8:0008      vpdFAT_FAT_size: b213
      vpi_ID: 25be2014         vpdFAT_dir_sector: fc04b800
      vpi_pDPB: #04c8:0038     vpdFAT_media: 0a
      vpi_cbSector: 0200       vpdFAT_next_free: 00b2
```

```

vpi_totsec: 00049020          vpdFAT_free_cnt: 04b8
vpi_trksec: 0023             vpdFAT_FATentrysize: b2
vpi_nhead: 000c             vpdFAT_IDsector: 00000000
vpi_pDCS: #0000:0000         vpdFAT_access: 0000
vpi_pVCS: #0000:0000         vpdFAT_accwait: 0000
vpi_drive: 02                vpdFAT_pEASFT: #0000:0000
vpi_unit: 02
vpi_text: UNLABELED
vpi_flags: 0003

```

#

```

>>> Two important pieces of information in the VPB: vpi_drive and
>>> vpi_text. The drive number is the logical drive, numbering from
>>> 0, Thus 02 is drive C:
>>> vpi_text is the volume label. in this case UNLABELED.
>>> The VPB contains a signature which when dumped as bytes appears as
>>> JD. Each VPB is 7b bytes, the first starts at +12. Each VPB can
>>> be dumped using the formula: 98:(12+(7b*entry))

```

```

# db 98:12+(7b*0) 17b
0098:00000012 00 00 8d 00 57 00 04 00-00 4a 44 02 00 08 00 c8 ....W....JD....H
0098:00000022 00 02 00 00 00 00 30 00-00 11 00 06 5c 7d 13 b2 .....0.....\}.2
0098:00000032 00 b8 04 fc 0a b2 00 b8-04 b2 00 00 00 00 0a 11 .8.|.2.8.2.....
0098:00000042 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
0098:00000052 00 00 00 00 00 00 00 00-00 00 00 9a 7b 4c 5c 00 .....{L\..
0098:00000062 00 14 20 be 25 38 00 c8-04 00 02 20 90 04 00 23 .. >%8.H... ..#
0098:00000072 00 0c 00 55 4e 4c 41 42-45 4c 45 44 00 00 00 00 ...UNLABELED....
0098:00000082 00 00 00 00 00 00 00 02-02 03 00 .....

```

```

>>> The word at +2 is a chain pointer offset to the next VPB. In this case
>>> 008d (=7b+12)

```

```

>>> We can also obtain a link to disk device driver information from
>>> the VPB via vpi_pDPB (the disk parameter block). Under the
>>> kernel debugger this may be formatted using .D DPB, but gives
>>> erroneous results under DF.

```

```

##.d dpb 4c8:38
      dpb_drive: 02
      dpb_unit: 02
dpb_driver_addr: #0738:0000
      dpb_next_dpb: #04c8:0054
      dpb_cbSector: 0200
      dpb_first_FAT: 0001
dpb_toggle_time: 00000000
      dpb_hVPB: 0012
      dpb_media: f8
      dpb_flags: 20
      dpb_drive_lock: 0000
      dpb_strategy2: #0740:135e

```

```

>>> From here we can locate the device driver header, but note that the
>>> strategy2 routine address is located from the DPB.

```

```

##.d dev 738:0
      DevNext: 0588:0000
      DevAttr: 2880
      DevStrat: 0d7e
      DevInt: 0000
      NumUnits: 08
      DevProtCS: 0740
      DevProtDS: 0738
      DevRealCS: 0000
      DevRealDS: 0000

```

```

>>> Returning to the JFN_table for slot 36. We now examine JFN 0004
>>> which correlates with SFN 002a
>>> Dump the SFT as before:

```

```

# .d sft 438:(8+(83*2a))
      sf_ref_count: 000c          sfi_mode: 0042
      sf_usercnt: 0000          sfi_hVPB: 0000
      reserved: 00             sfi_ctime: 0000
      sf_flags(2): 00c1:0000     sfi_cdate: 0000
      sf_devptr: #04f8:0000      sfi_atime: 0000
      sf_FSC: #00c8:ff40         sfi_adata: 0000
      sf_chain: #0438:0f62       sfi_mtime: 3c83
      sf_MFT: fe73ecfc          sfi_mdate: 1d62

```

```

sfdFAT_firFILEclus: 0000          sfi_size: 00000000
sfdFAT_cluspos: 0000             sfi_position: 00000000
sfdFAT_lstclus: 0000             sfi_UID: 0000
sfdFAT_dirsec: 00000000          sfi_PID: 0003
sfdFAT_dirpos: 00                sfi_PDB: 0000
sfdFAT_name: KBD$                sfi_selfsfn: 002a
sfdFAT_EAHandle: 0000            sfi_tstamp: 00
sf_plock: 0000                   sfi_DOSattr: 00
sf_NmPipeSfn: 0000
sf_codepage: 0000

>>> The first flag word is 00c1 = 0000 0000 1100 0001
>>>
>>>
>>> Device ..
>>>
>>> ..
>>> .console input dev
>>>
>>> .
>>> not console output dev
>>> Note: the hVPB is 0000 but sf_devptr is not and points to the
>>> device driver header for KBD$ thus:

##.d dev 4f8:0
    DevNext: 04e8:0000
    DevAttr: c981
    DevStrat: 0000
    DevInt: 2a29
    DevName: KBD$
    DevProtCS: 0500
    DevProtDS: 04f8
    DevRealCS: 0000
    DevRealDS: 0000

>>> The MFT entry for this device is:
# db %fe73ecfc
%fe73ecfc 4b 53 45 4d 01 02 00 00-00 00 00 00 00 00 00 00 KSEM.....
%fe73ed0c 00 00 33 1d 38 04 00 00-00 00 4b 53 45 4d 01 02 ..3.8.....KSEM..
%fe73ed1c 00 00 00 00 00 00 00 00 00-00 00 00 00 2a 00 00 00 .....*...
%fe73ed2c 6d 46 00 00 5c 44 45 56-5c 4b 42 44 24 00 73 fe mF..\DEV\KBD$.s~
%fe73ed3c 30 00 a6 ff 02 00 f9 00-48 2f 1c fd 60 ed 73 fe 0.&...y.H/.}'ms~
%fe73ed4c 94 ed 73 fe 88 b1 98 04-01 00 00 00 1f 00 00 00 .ms~.1.....
%fe73ed5c a4 dc 72 fe 08 42 4d 53-43 41 4c 4c 53 ec 73 fe $\r~.BMSCALLSls~
%fe73ed6c 18 00 9e ff 3c 00 0b 00-d4 ef 73 fe cc 14 86 fe ....<...Tos~L..~
#
>>> Note: the name of the KBD$ device driver known to the file system is
>>> \DEV\KBD$

```

```

>>> Finally, using the .D MFT command (under the KDB) this MFT formats as:

##.d mft %fe73ecfc
    mft_ksem:
Signature      : KSEM                      Nest: 0000
Type           : SHARE                     Readers: 0000
Flags          : 01                       PendingReaders: 0000
Owner          : 0000                     PendingWriters: 0000
    mft_lptr: 0000                        mft_sptr: 0438:1586
    mft_pCMap: 00000000                   mft_serl: 002c      mft_signature: 466d
    mft_CMapKSem:
    mft_hvpb: 0000                        mft_opflags: 0000      mft_flags: 0000
    mft_name: \DEV\KBD$
>>> Note: mft_sptr points to the associated SFT

```

## Finding Files From Handles in a VDM

The situation in a VDM is slightly more complex, since it required the **JFN** to be compatible with **DOS** and therefore an 8-bit entity.

Furthermore, the **JFN\_table** in DOS is traditionally imbedded or chained from the DOS **PDB (or PSP)**. For this reason a second level of indirection is employed.

The **JFN** returned from a VDM open indexes the byte array of virtual system file number (**VSFNs**). The **VSFN** ranges from 0 - 255. The high 47 (from 0xd0 though 0xfe) are used as real mode device handles. 0xff indicates an unused handle. When a VDM is created the initial **PDB** contains the default array of 20 handles at label **PDB\_JFN\_table (PDB + 0x18)**. This current array's far segment address is at **PDB\_JFN\_pointer (PDB + 0x34)** and the size of the array is a word at **PDB\_JFN\_Length (PDB + 0x32)**. The **PDB** lies on a paragraph boundary (16-byte boundary) and its segment address is saved in the **PTDA** at **CurrentPDB (PTDA + 0x2ea)** Once again the usual precaution applies when referencing **PTDA** fields: their symbols are publicly defined for the current system context only. Therefore, to reference a **CurrentPDB** out-of-context must be done relative to the **PTDA** address for that context.

These points are illustrated in the following example:

```
##.p 46
  Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
  0046  001d  0007  001d  0001  blk  0200  7b732000  7b8c9a04  7b8af720  1f08  17  *vdm
##.s 46

>>> Slot 46 is a VDM but not the current context, so we locate the PDB
>>> relative to the PTDA (otherwise we could have just used
>>> dw currentpdb 11)

##dw %7b8c9a04+currentpdb-ptda_start 11
0030:0000fd16  0e01

>>> This is a segment address so use the & operator to display the
>>> PDB.

##db &e01:0
&0e01:00000000  cd 20 00 a0 00 9a f0 fe-1d f0 f5 01 99 0d 08 02 M . .p~.pu....
&0e01:00000010  28 08 65 07 28 08 99 0d-d1 d1 d1 d0 d2 ff ff ff (.e.(...QQQPR...
&0e01:00000020  ff ff ff ff ff ff ff ff-ff ff ff ff d7 00 e4 03 .....W.d.
&0e01:00000030  11 0e 30 00 00 00 00 40 09-ff ff ff ff 00 00 00 00 ..0...@.....
&0e01:00000040  14 0b 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
&0e01:00000050  cd 21 cb 72 6e 6c 20 2d-20 4e 6f 74 00 20 20 20 M!Krn1 - Not.
&0e01:00000060  20 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20 .....
&0e01:00000070  20 20 20 20 20 20 20 20-00 00 00 00 e7 08 50 03 ....g.P.

>>> Word at PDB+0x32 is 0030. This is the number of file handles
>>> supported in this VDM. The default is 0014. So the PDB_JFN_table
>>> has been expanded.
>>> Far pointer at PDB+34 is &0940:0000. This is the current
>>> PDB_JFN_table address. (By default this would have pointed to
>>> PDB+0x18, but the table has been expanded.)

>>> Now dump the current PDB_JFN_table.

##db &940:0
&0940:00000000  d1 d1 d1 d0 d2 00 01 02-03 04 05 06 07 08 09 ff QQQPR.....
&0940:00000010  0b ff ff ff ff ff ff ff-ff ff ff ff ff ff ff .....
&0940:00000020  ff ff ff ff ff ff ff ff-ff ff ff ff ff ff ff .....
&0940:00000030  4d 00 00 08 00 00 00 00-00 00 00 00 00 00 00 M.....
&0940:00000040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
&0940:00000050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
&0940:00000060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
&0940:00000070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....

>>> JFNs 0 - 4 correspond to VSFNs d1, d1, d1, d0 and d2. Each of these
>>> is greater than 0xcf and therefore a real mode device handle
>>> and not managed by the protect mode file system.
>>> JFN 5 is the first open file in this VDM. It has VSFN 00. This
>>> may be used as an index into the protect mode JFN_table to
>>> find the corresponding SFT, MFT and file name. The technique from
>>> this point is the same as in the preceding section.

>>> Dump the JFN_table from the PTDA.

##dw %7b8c9a04 jfn_ptable-ptda_start 12
0030:0000ffbc  0000 lea8

>>> We are no longer based on selector 30 so the JFN table has been
>>> expanded. Now dump the current table...

##dw #lea8:0
lea8:00000000  006a 0069 0075 008c 008b 005e 0089 0088
```

```

lea8:00000010  008a 0097 ffff 008d ffff ffff ffff ffff
lea8:00000020  ffff ffff ffff ffff ffff ffff ffff ffff
lea8:00000030  ffff ffff ffff ffff ffff ffff ffff ffff
lea8:00000040  ffff ffff ffff ffff ffff ffff ffff ffff
lea8:00000050  ffff ffff ffff ffff ffff ffff ffff ffff
lea8:00000060
Past end of segment: lea8:00000060

>>> VSFN 00 corresponds to SFN 006a. Now dump the SFT...

##.d sft 438:(8+(83*6a))
      sf_ref_count: 0001                sfi_mode: 00a0
      sf_usercnt: 0000                sfi_hVPB: 0012
      reserved: 00                sfi_ctime: 0000
      sf_flags(2): 0000:0000        sfi_cdate: 0000
      sf_devptr: #0000:0000        sfi_atime: 0000
      sf_FSC: #00c8:0008          sfi_adate: 0000
      sf_chain: #0000:0000        sfi_mtime: 0000
      sf_MFT: fe7c8b54            sfi_mdate: 0000
sfdFAT_firFILEclus: 4d58            sfi_size: 00000594
      sfTrap 13 (0DH) - General Protection Fault 0000 - In Debugger
eax=90000000 ebx=ffdd55ba ecx=00000000 edx=013c0000 esi=00006373 edi=ffff2940
eip=000054ae esp=00003b0a ebp=00003b14 iopl=0         rf -- -- nv up di pl zr na pe nc
cs=0120  ss=0128  ds=0128  es=0128  fs=0168  gs=0000             cr2=16ea2000  cr3=001d9000
0120:000054ae ff56fe          call     word ptr [bp-02]          ss:3b12=b23b

>>> Oops! the debugger had a problem. Not to worry, he told us
>>> the MFT address, so dump that ... (this also appears at SFT+0x19)

>>> the SFT again ...
##dw 438:(8+(83*6a))
0438:00003646  0001 0000 0000 0000 0000 0000 0800 c800
0438:00003656  0000 0000 0000 0000 5400 7c8b 58fe 484d
0438:00003666  001f 0000 0000 0000 0000 0000 0000 0000
0438:00003676  0000 0000 0000 0000 0000 0000 0000 0000
0438:00003686  0000 7830 007a 0000 0000 0000 0000 a000
0438:00003696  0000 1200 0000 0000 0000 0000 0000 0000
0438:000036a6  9400 0005 9400 0005 0000 1d00 0100 6a0e
0438:000036b6  0000 0000 0000 0000 0020 0000 0000 0000

>>> MFT
##db %fe7c8b54
%fe7c8b54 4b 53 45 4d 01 02 00 00-00 00 00 00 00 00 00 00 KSEM.....
%fe7c8b64 00 00 46 36 38 04 00 00-00 00 4b 53 45 4d 01 02 ..F68....KSEM..
%fe7c8b74 00 00 00 00 00 00 00 00-00 00 00 00 ba 0f 00 00 .....:...
%fe7c8b84 6d 46 12 00 43 3a 5c 4f-53 32 5c 4d 44 4f 53 5c mF...C:\OS2\MDOS\
%fe7c8b94 57 49 4e 4f 53 32 5c 53-59 53 54 45 4d 5c 4b 42 WINOS2\SYSTEM\KB
%fe7c8ba4 44 55 4b 2e 44 4c 4c 00-52 8b 7c fe 3c 00 7d ff DUK.DLL.R.|~<.).
%fe7c8bb4 00 00 00 00 00 00 00 00-f0 8b 7c fe 20 f7 8a 7b .....p.|~ w.{
%fe7c8bc4 f0 f1 f9 78 00 04 00 00-48 4f 4f 4b 60 bf f9 ff pqyx....HOOK'?y.

>>> The file name is: C:\OS2\MDOS\WINOS2\SYSTEM\KBDUK.DLL

```

## Finding Handles From File Names

File system names are recorded in the **MFT** control block. Each **MFT** has a handle, which is its linear address and a key which is the concatenation of the **hVPB** with the file name considered as a string of bytes. The **MFT** keys are managed in a **Patricia Tree** structure similar to that described by *Knuth* in *The Art of computer Programming, Volume 3, Sorting and Searching Algorithms*. However, note that the implementation of the **PTree** in OS/2 is slightly modified from the *Knuth* treatment.

The **SAS** gives us the address of the header node for the **MFT PTree**. The header node points to the first **PTree** entry. Each entry comprises a bit index, a key length, a left pointer, a right pointer and an **MFT** handle. The bit-index is used to specify the bit in the key to be tested. If the bit 0 then the left pointer is taken, otherwise the right pointer is taken. When the selected pointer points back to the same **PTree** entry then the search stops and the required **MFT** is found from the **MFT** handle. The bit index count the bits of each byte of the key from left to right.

This technique is now illustrated in the following example:

>>> Who's got C:\OS2\HELP\HMHELP.HLP open?

>>> First look through VPBs to find hVPB for C:

>>> Find the VPB segment and chain through them starting with the  
>>> first at offset 0x12.

```
# ln gdt_vpb
0138:00000098 OS2KRNL GDT_VPB
# .d vpb 98:12
    vpb_flink: 0000                vpdFAT_cluster_mask: 02
    vpb_blink: 008d                vpdFAT_cluster_shift: 00
    vpb_ref_count: 0057            vpdFAT_first_FAT: 0000
    vpb_search_count: 0004         vpdFAT_FAT_count: 00
    vpb_first_access: 00           vpdFAT_root_entries: 0030
    vpb_signature: 444a            vpdFAT_first_sector: 06001100
    vpb_flags(2): 02:00            vpdFAT_max_cluster: 7d5c
    vpb_FSC: #00c8:0008            vpdFAT_FAT_size: b213
    vpi_ID: 25be2014               vpdFAT_dir_sector: fc04b800
    vpi_pDPB: #04c8:0038           vpdFAT_media: 0a
    vpi_cbSector: 0200             vpdFAT_next_free: 00b2
    vpi_totsec: 00049020           vpdFAT_free_cnt: 04b8
    vpi_trksec: 0023               vpdFAT_FATentrysize: b2
    vpi_nhead: 000c                vpdFAT_IDsector: 00000000
    vpi_pDCS: #0000:0000           vpdFAT_access: 0000
    vpi_pVCS: #0000:0000           vpdFAT_accwait: 0000
    vpi_drive: 02                  vpdFAT_pEASFT: #0000:0000
    vpi_unit: 02
    vpi_text: UNLABELED
    vpi_flags: 0003
```

>>> We get lucky the first time. This VPD is for drive 2, that is C:

>>> So the hVPB=0012 (i.e the offset into the VPD segment).

>>> We now need to form the MFT key for the file name we wish to

>>> look up. So convert the file name to ASCII and concatenate to the

>>> hVPB (as a byte pair, that is, reversed)

```
>>> C : \ O S 2 \ H E L P \ H M H E L P . H L P
>>> 12 00 43 3a 5c 4f 53 32 5c 48 45 4c 50 5c 48 4d 48 45 4c 50 2e 48-4c 50
```

>>> Locate the MFT PTree head from the SAS - in a dump use .A

>>> otherwise unravel the SAS from selector 70

```
# .a
--- SAS Base Section ---
    SAS signature: SAS
    offset to tables section: 0016
    FLAT selector for kernel data: 0168
    offset to configuration section: 001E
    offset to device driver section: 0020
    offset to Virtual Memory section: 002C
    offset to Tasking section: 005C
    offset to RAS section: 006E
    offset to File System section: 0074
    offset to infoseg section: 0080
--- SAS Protected Modes Tables Section ---
    selector for GDT: 0008
    selector for LDT: 0000
    selector for IDT: 0018
    selector for GDTPOOL: 0100
--- SAS Device Driver Section ---
    offset for the first bimodal dd: 0CB9
    offset for the first real mode dd: 0000
    sel for Drive Parameter Block: 04C8
    sel for ABIOS prot. mode CDA: 0000
    seg for ABIOS real mode CDA: 0000
    selector for FSC: 00C8
--- SAS Task Section ---
    selector for current PTDA: 0030
    FLAT offset for process tree head: FFF10910
    FLAT address for TCB address array: FFF06BB6
    offset for current TCB number: FFDFFB5E
    offset for ThreadCount: FFDFFB62
--- SAS File System Section ---
    handle to MFT PTree: FE72CFBC
    selector for System File Table: 00C0
    sel. for Volume Parameter Bloc: 0788
    sel. for Current Directory Struc: 07B8
    selector for buffer segment: 00A8
```

```

--- SAS Information Segment Section ---
    selector for global info seg: 0428
    address of curtask local infoseg: 03C80000
    address of DOS task's infoseg: FFFFFFFF
    selector for Codepage Data: 07CB
--- SAS RAS Section ---
selector for System Trace Data Area: 04B0
segment for System Trace Data Area: 04B0
offset for trace event mask: 0B28
--- SAS Configuration Section ---
offset for Device Config. Table: 0D50
--- SAS Virtual Memory Mgt. Section ---
    Flat offset of arena records: FFF13304
    Flat offset of object records: FFF1331C
    Flat offset of context records: FFF1330C
    Flat offset of kernel mte records: FFF0A891
    Flat offset of linked mte list: FFF07934
    Flat offset of page frame table: FFF11A70
    Flat offset of page range table: FFF111EC
    Flat offset of swap frame array: FFF03BAC
        Flat offset of Idle Head: FFF10090
        Flat offset of Free Head: FFF10080
        Flat offset of Heap Array: FFF11B78
    Flat offset of all mte records: FFF12E04

>>> MFT Ptree is at %fe72cfbc
>>> each entry including the header has the following format:
>>> +0 W Bit index
>>> +2 W key length
>>> +4 D left link
>>> +8 D right link
>>> +c D MFT handle (MFT pointer)

>>> dump the header and the first entry pointed to by the left link
# dd %FE72CFBC l4
%fe72cfbc ffffffff fe867f10 fe72cfbc 00000000
# dd %FE867f10 l4
%fe867f10 00100000 fe861454 fe861470 fe721a04
>>>      ----.... -----
>>>      Kl  BI    left    right    MFT
>>>
>>> Note the word reversal of the Bit index and the Key length because
>>> we dumped double-words.
>>> BI tells us to test bit 0 of the key (numbering from the left
>>> starting with 0). 12 00 .. .. = 0001 0010 0000 0000 ....
>>> Bit zero is 0 so take the left link.

# dd %FE861454 l4
%fe861454 00100001 fe73d194 fe845370 fe72196c

>>> Now test bit 1. This is still zero. Again take the left link.

# dd %FE73d194 l4
%fe73d194 00190003 fe72cf3c fe87ec34 fe72dea4

>>> Now test bit 3. This is 1 so take the right link.

# dd %FE87ec34 l4
%fe87ec34 000b0029 fe87ebdc fe834308 fe87ebf0

>>> Now test bit 29. .... 4f .... = 0100 1111
>>> This is 1 so take the right link.

# dd %FE834308 l4
%fe834308 0019002b fe869f30 fe834254 fe834274

>>> Test bit 2b. This is 0. Turn left.

# dd %FE869f30 l4
%fe869f30 0017002c fe885ac4 fe87ec90 fe869ee0

>>> Test bit 2c. This is 1. Turn right.

# dd %FE87ec90 l4
%fe87ec90 000f002d fe87ec90 fe834ac8 fe87ec48

>>> Test bit 2d. This is 1. Turn right.

# dd %FE834ac8 l4

```



```
%fe834ac8 001a0044 fe845ef8 fe724fe4 fe845c0c
```

```
>>> Test bit 44. ....5c.... = 0101 1100.
```

```
>>> This is 1. Turn right.
```

```
# dd %FE724fe4 l4
```

```
%fe724fe4 001b004b fe801414 fe862de8 fe722fac
```

```
>>> Test bit 4b. ....48..... = 0100 1000
```

```
>>> This is 0. Turn left.
```

```
# dd %FE801414 l4
```

```
%fe801414 0017004c fe801d90 fe7cef84 fe7dffb0
```

```
>>> Test bit 4c. This is 1. Turn right.
```

```
# dd %FE7cef84 l4
```

```
%fe7cef84 00180073 fe7cef84 fe801414 fe7cef30
```

```
>>> Test bit 73. ....48.... = 0100 1000
```

```
>>> This is zero and the left link points to the same node.
```

```
>>> Therefore the search ends and we should have found the MFT
```

```
>>> for our file name. Dump the MFT to check .....
```

```
# db %FE7cef30 l50
```

```
%fe7cef30 4b 53 45 4d 01 02 00 00-00 00 00 00 00 00 00 00 KSEM.....  
%fe7cef40 00 00 28 31 38 04 00 00-00 00 4b 53 45 4d 01 02 ..(18.....KSEM..  
%fe7cef50 00 00 00 00 00 00 00 00-00 00 00 00 69 03 00 00 .....i...  
%fe7cef60 6d 46 12 00 43 3a 5c 4f-53 32 5c 48 45 4c 50 5c mF..C:\OS2\HELP\  
%fe7cef70 48 4d 48 45 4c 50 2e 48-4c 50 00 00 16 e6 7c fe HMHELP.HLP...f|~
```

```
>>> The MFT + 0x22 points is the SFT segment's offset. So dump the
```

```
>>> SFT .....
```

```
# ln gdt_sft
```

```
0138:000000c0 OS2KRNL GDT_SFT
```

```
dw c0:011
```

```
#00c0:00000000 0438
```

```
# .d sft 438:3128
```

```
sf_ref_count: 0001 sfi_mode: 00a0  
sf_usercnt: 0000 sfi_hVPB: 0012  
reserved: 00 sfi_ctime: 0000  
sf_flags(2): 0000:0000 sfi_cdate: 0000  
sf_devptr: #0000:0000 sfi_atime: 0000  
sf_FSC: #00c8:0008 sfi_adate: 0000  
sf_chain: #0438:33b7 sfi_mtime: 0000  
sf_MFT: fe7cef30 sfi_mdate: 0000  
sfdFAT_firFILEclus: 3344 sfi_size: 00007058  
sfdFAT_cluspos: 0f10 sfi_position: 00000a90  
sfdFAT_lstclus: 0000 sfi_UID: 0000  
sfdFAT_dirsec: 00000000 sfi_PID: 0012  
sfdFAT_dirpos: 00 sfi_PDB: 0000  
sfdFAT_name: sfi_selfsfn: 0060  
sfdFAT_EAHandle: 0000 sfi_tstamp: 00  
sf_plock: 0000 sfi_DOSattr: 00  
sf_NmPipeSfn: 0000  
sf_codepage: 0000
```

```
>>> sfi_PID tells us PID 12 has opened this file. But the
```

```
>>> sf_chain is not zero, so other processes have also opened
```

```
>>> this file. Follow the sf_chain .....
```

```
# .d sft 438:33b7
```

```
sf_ref_count: 0001 sfi_mode: 00a0  
sf_usercnt: 0000 sfi_hVPB: 0012  
reserved: 00 sfi_ctime: 0000  
sf_flags(2): 0000:0000 sfi_cdate: 0000  
sf_devptr: #0000:0000 sfi_atime: 0000  
sf_FSC: #00c8:0008 sfi_adate: 0000  
sf_chain: #0438:2d10 sfi_mtime: 0000  
sf_MFT: fe7cef30 sfi_mdate: 0000  
sfdFAT_firFILEclus: 284a sfi_size: 00007058  
sfdFAT_cluspos: 0f10 sfi_position: 00000a90  
sfdFAT_lstclus: 0000 sfi_UID: 0000  
sfdFAT_dirsec: 00000000 sfi_PID: 000c  
sfdFAT_dirpos: 00 sfi_PDB: 0000  
sfdFAT_name: sfi_selfsfn: 0065
```

```

sfdFAT_EAHandle: 0000          sfi_tstamp: 00
sf_plock: 0000               sfi_DOSattr: 00
sf_NmPipeSfn: 0000
sf_codepage: 0000

# .d sft 438:2d10
sf_ref_count: 0001           sfi_mode: 00a0
sf_usercnt: 0000             sfi_hVPB: 0012
reserved: 00                 sfi_ctime: 0000
sf_flags(2): 0000:0000       sfi_cdate: 0000
sf_devptr: #0000:0000        sfi_atime: 0000
sf_FSC: #00c8:0008          sfi_adate: 0000
sf_chain: #0438:2e16         sfi_mtime: 0000
sf_MFT: fe7cef30            sfi_mdate: 0000
sfdFAT_firFILEclus: 1986     sfi_size: 00007058
sfdFAT_cluspos: 0f10         sfi_position: 00000a90
sfdFAT_lstclus: 0000         sfi_UID: 0000
sfdFAT_dirsec: 00000000      sfi_PID: 000b
sfdFAT_dirpos: 00           sfi_PDB: 0000
sfdFAT_name:                 sfi_selfsfn: 0058
sfdFAT_EAHandle: 0000        sfi_tstamp: 00
sf_plock: 0000              sfi_DOSattr: 00
sf_NmPipeSfn: 0000
sf_codepage: 0000

# .d sft 438:2e16
sf_ref_count: 0001           sfi_mode: 00a0
sf_usercnt: 0000             sfi_hVPB: 0012
reserved: 00                 sfi_ctime: 0000
sf_flags(2): 0000:0000       sfi_cdate: 0000
sf_devptr: #0000:0000        sfi_atime: 0000
sf_FSC: #00c8:0008          sfi_adate: 0000
sf_chain: #0000:0000         sfi_mtime: 3ca2
sf_MFT: fe7cef30            sfi_mdate: 1d62
sfdFAT_firFILEclus: 050c     sfi_size: 00007058
sfdFAT_cluspos: 0f10         sfi_position: 00000a90
sfdFAT_lstclus: 0000         sfi_UID: 0000
sfdFAT_dirsec: 00000000      sfi_PID: 000d
sfdFAT_dirpos: 00           sfi_PDB: 0000
sfdFAT_name: SINGLEQ$        sfi_selfsfn: 005a
sfdFAT_EAHandle: 0000        sfi_tstamp: 00
sf_plock: 0000              sfi_DOSattr: 00
sf_NmPipeSfn: 0000
sf_codepage: 0000

```

```

>>> In all, PIDs 0012, 000c, 000b and 000d have opened
>>> C:\OS2\HELP\HMHELP.HLP

```

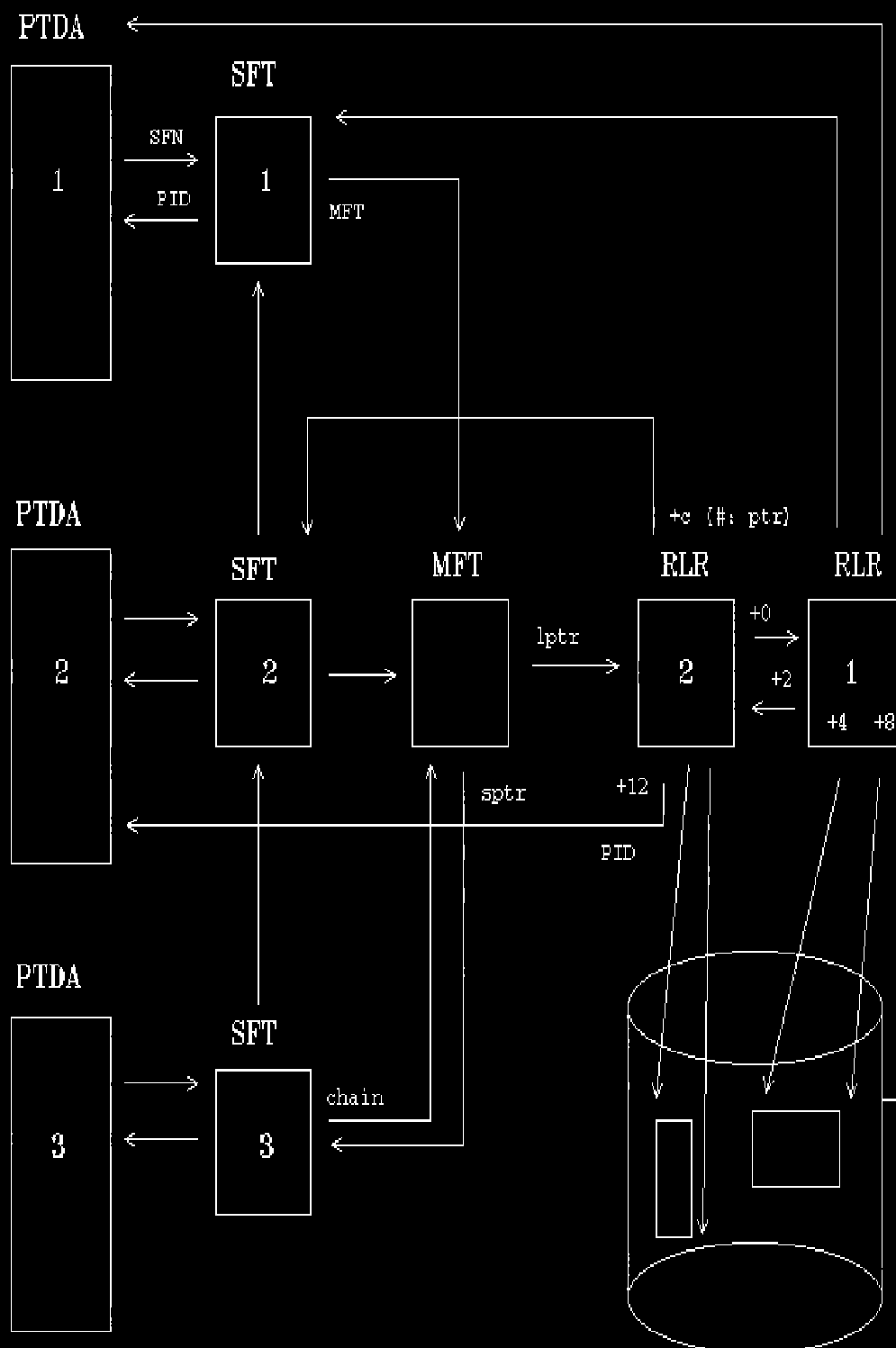
## The Record Lock Record

In this example we investigate the [RLR](#) and how it records a locked range within a file. We will also see how the Block-Id of a thread waiting for access to a locked file range directly leads to discovery of the RLR.

We introduce the **RLR** by showing its relationship to other file-system control blocks in the following diagram. This depicts the following situation:

- Three processes have opened the same file, in the order process 1, 2, then 3.
- The **MFT** heads the chain of **SFTs**, each representing an open instance of the same file. The **MFT** points to the most recent **SFT** open instance.
- Process 1 and process 2 have each locked a range within the same file. **RLRs** 1 and 2 correspond to process 1 and 2.
- The **MFT** heads the chain of **RLRs** starting with the most recent. The pointer from the **MFT** (**lptr**) is the offset within the **RLR** segment.

# Shared File with 2 Locked Ranges



# A Hang Problem Involving Locked Records

```
>>> Problem: Program "Pain" running in Slot 4d is hung with a blank
>>> screen. Everything else in the system seems OK. Mouse moves, we
>>> can change focus and so on..

>>> Lets take a look at slot 4d.

# .p 4d
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
004d# 001c 001b 001c 0001 blk 0300 ab80f000 ab99e220 ab980620 1e60 1d PAIN

>>> Blocked!

>>> We can approach this two ways:

>>> 1) Take a look at what the application did

>>> 2) Take a look at the BlockId and try see how far the system got

>>> Looking at the application we examine its registers and determine
>>> what API it called to cause it to block.

# .s 4d
Current slot number: 004d

# .r
eax=00023305 ebx=00000000 ecx=00000006 edx=0002004f esi=00000000 edi=00000000
eip=00010181 esp=0002337c ebp=000233fc iopl=2 -- -- -- nv up ei pl zr na pe nc
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001d9000
005b:00010181 83c414 add esp,+14

# u %eip-10

%00010171 e8508d45e0 call %e0468ec6
%00010176 50 push eax
%00010177 ff75f8 push dword ptr [ebp-08]
%0001017a b005 mov al,05
%0001017c e8c762f81b call %1bf96448
%00010181 83c414 add esp,+14
%00010184 0bc0 or eax,eax
%00010186 7404 jz %0001018c
%00010188 ebde jmp %00010168
%0001018a 8bc0 mov eax,eax
%0001018c b858000200 mov eax,00020058
%00010191 e8fa000000 call %00010290
# ln %1bf96448

%1bf96448 DOSCALL1 DOS32SETFILELOCKS

>>> The last call was to DosSetFileLocks and we haven't returned. If
>>> we want any more information we have to analyse the BlockId.

# .pb#

Slot Sta BlockID Name Type Addr Symbol
004d# blk 00b80029 PAIN
# .m 0b8:29

*har par cpg va flg next prev link hash hob hal
00dd %feaf0308 00000010 %aa7a3000 129 00dc 00de 0000 00cb 00ea 0000 sel=00b8
hob har hobnxt flgs own hmte sown,cnt lt st xf
00ea 00dd 0000 0124 ff47 0000 0000 00 00 00 00 fsreclok
```

```
>>> Seems to be blocked on a File System Record Lock (RLR).
>>> This implies that someone else has already locked a
>>> conflicting record range.

>>> We need to dump the RLR and locate the System File Table Entry
>>> associated with it. The BlockId is the address of the RLR.
```

```
# dw 0b8:29
```

```
00b8:00000029 0000 0000 0020 0000 002f 0000 526b 00d0
00b8:00000039 0000 0018 0000 0002 0000 2000 0000 2f00
00b8:00000049 0000 2000 9806 29ab 1c00 0300 0000 006e
00b8:00000059 0000 0000 0000 0000 0000 0000 0000 0000
00b8:00000069 0000 0000 8500 0000 0000 0000 0000 0000
00b8:00000079 0000 0000 0000 0000 0000 0000 009c 0000
00b8:00000089 0000 0000 0000 0000 0000 0000 0000 0000
00b8:00000099 0000 b300 0000 0000 0000 0000 0000 0000
```

```
>>> RLR+c is a far16 pointer to the associated System File Table entry
>>> (SFT).
```

```
>>> RLR+4 and RLR+8 are the range of bytes locked. Offset +20 - +2f
>>> has been locked.
```

```
>>> We now format the SFT for the process that locked this range:
```

```
# .d sft d0:526b
```

```
sf_ref_count: 0001          sfi_mode: 0042
sf_usercnt: 0000           sfi_hVPB: 04e0
reserved: 00              sfi_ctime: 0000
sf_flags(2): 0040:0000    sfi_cdate: 0000
sf_devptr: #0000:04e0     sfi_atime: 0000
sf_FSC: #0000:ff40        sfi_adate: 0000
sf_chain: #0000:0000      sfi_mtime: 5df6
sf_MFT: fe87ca0c          sfi_mdate: 1f3a
sfdFAT_firFILEclus: 0197   sfi_size: 00000061
sfdFAT_cluspos: 0000       sfi_position: 00000030
sfdFAT_lstclus: 0197       sfi_UID: 0000
sfdFAT_dirsec: 0000009f    sfi_PID: 0018
sfdFAT_dirpos: 0a          sfi_PDB: 0000
sfdFAT_name: VIN           sfi_selfsfid: 00a1
sfdFAT_EAHandle: 0000      sfi_tstamp: 00
sf_plock: 0000             sfi_DOSattr: 20
sf_NmPipeSfn: 0000
sf_codepage: 0000
```

```
>>> The SFT contains a pointer to the MFT, which contains the fully
>>> qualified file name. If the file is FAT then the short name in the
>>> SFT is also meaningful.
```

```
# .d mft % fe87ca0c
```

```
mft_ksem:
Signature      : KSEM          Nest: 0000
Type           : SHARE        Readers: 0000
Flags          : 01           PendingReaders: 0000
Owner          : 0000         PendingWriters: 0000
mft_lptr: 0029             mft_sptr: 00d0:5600
mft_pCMap: 00000000        mft_serl: 128f
mft_CMapKSem:
mft_hvpb: 466d             mft_opflags: 0000      mft_flags: 0000
mft_name: A:\LAB19\VIN
```

```
>>> So the locked File is a:\lab19\vin
```

```
>>> The SFT also contains the Pid of the process that opened, and in
>>> this case locked, this file - Pid 18
```

```
# .p
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
0001  0001  0000  0000  0001  blk  0100  ffe3a000  ffe3ca00  ffe3c800  1e7c  00  *ager
0002  0001  0000  0000  0002  blk  0200  ab779000  ffe3ca00  ab977020  1f3c  00  *tsd
0003  0001  0000  0000  0003  blk  0200  ab77b000  ffe3ca00  ab977220  1f50  00  *ctxh
```

0004	0001	0000	0000	0004	blk	081f	ab77d000	ffe3ca00	ab977420	1f48	00	*kdb
0005	0001	0000	0000	0005	blk	0800	ab77f000	ffe3ca00	ab977620	1f20	00	*lazyw
0006	0001	0000	0000	0006	blk	0800	ab781000	ffe3ca00	ab977820	1f3c	00	*asynchr
0009	0002	0000	0002	0001	rdy	0804	ab787000	ab997020	ab977e20	1c88	00	CNTRL
0008	0002	0000	0002	0002	blk	0804	ab785000	ab997020	ab977c20		00	CNTRL
000b	0002	0000	0002	0003	blk	0804	ab78b000	ab997020	ab978220		00	CNTRL
000c	0002	0000	0002	0004	rdy	0804	ab78d000	ab997020	ab978420	1c9c	00	CNTRL
000a	0003	0000	0003	0001	blk	0800	ab789000	ab997620	ab978020		00	DOSCTL
000d	0004	0001	0004	0001	rdy	0500	ab78f000	ab997c20	ab978620	1ed0	01	PMSHL32
000f	0004	0001	0004	0002	blk	0800	ab793000	ab997c20	ab978a20	1ed4	01	PMSHL32
0010	0004	0001	0004	0003	blk	0800	ab795000	ab997c20	ab978c20		01	PMSHL32
0011	0004	0001	0004	0004	blk	0800	ab797000	ab997c20	ab978e20		01	PMSHL32
0012	0004	0001	0004	0005	blk	0800	ab799000	ab997c20	ab979020		01	PMSHL32
0015	0004	0001	0004	0006	blk	0200	ab79f000	ab997c20	ab979620	1edc	01	PMSHL32
0016	0004	0001	0004	0007	blk	0200	ab7a1000	ab997c20	ab979820	1edc	01	PMSHL32
0017	0004	0001	0004	0008	blk	0200	ab7a3000	ab997c20	ab979a20		01	PMSHL32
0007	0004	0001	0004	0009	blk	0500	ab783000	ab997c20	ab977a20		01	PMSHL32
0018	0004	0001	0004	000a	blk	0800	ab7a5000	ab997c20	ab979c20		01	PMSHL32
0019	0004	0001	0004	000b	blk	0800	ab7a7000	ab997c20	ab979e20	1eb8	01	PMSHL32
001a	0004	0001	0004	000c	blk	0800	ab7a9000	ab997c20	ab97a020	1eb8	01	PMSHL32
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pTDA	pTCB	Disp	SG	Name
001b	0004	0001	0004	000d	blk	0804	ab7ab000	ab997c20	ab97a220	1ea8	01	PMSHL32
001c	0004	0001	0004	000e	blk	0804	ab7ad000	ab997c20	ab97a420	1eb0	01	PMSHL32
001d	0004	0001	0004	000f	blk	0500	ab7af000	ab997c20	ab97a620	1ea8	01	PMSHL32
001e	0004	0001	0004	0010	blk	0801	ab7b1000	ab997c20	ab97a820	1bac	01	PMSHL32
001f	0004	0001	0004	0011	blk	0801	ab7b3000	ab997c20	ab97aa20		01	PMSHL32
0020	0004	0001	0004	0012	blk	0801	ab7b5000	ab997c20	ab97ac20		01	PMSHL32
0021	0004	0001	0004	0013	blk	0800	ab7b7000	ab997c20	ab97ae20		01	PMSHL32
0022	0004	0001	0004	0014	blk	0800	ab7b9000	ab997c20	ab97b020	1b80	01	PMSHL32
0024	0004	0001	0004	0015	blk	0200	ab7bd000	ab997c20	ab97b420	1ed0	01	PMSHL32
0030	0004	0001	0004	0016	blk	0800	ab7d5000	ab997c20	ab97cc20	1eac	01	PMSHL32
004c	001b	0004	001b	0001	blk	0400	ab80d000	ab99dc20	ab980420	1ed4	1d	CMD
004b	001a	0004	001a	0001	blk	0200	ab80b000	ab99d620	ab980220	1eb8	1c	CMD
004a	0019	0004	0019	0001	blk	0200	ab809000	ab99d020	ab980020	1eb8	14	CMD
0048	0017	0004	0017	0001	blk	0200	ab805000	ab99a620	ab97fc20	1ed4	04	CMD
0047	0014	0004	0014	0001	blk	0200	ab803000	ab99c420	ab97fa20	1eb8	11	CMD
003d	0012	0004	0012	0001	blk	0200	ab7ef000	ab99be20	ab97e620	1ed0	1a	IBMAVSD
0046	0011	0004	0011	0001	blk	0200	ab801000	ab99b820	ab97f820	1ed0	19	FPWMON
003a	0010	0004	0010	0001	blk	0200	ab7e9000	ab99b220	ab97e020	1ed0	18	PMFAX
0041	0010	0004	0010	0002	blk	0800	ab7f7000	ab99b220	ab97ee20	1edc	18	PMFAX
0043	0010	0004	0010	0003	blk	0500	ab7fb000	ab99b220	ab97f220		18	PMFAX
0045	0010	0004	0010	0005	blk	0500	ab7ff000	ab99b220	ab97f620	1d24	18	PMFAX
0039	000f	0004	000f	0001	blk	0200	ab7e7000	ab99ac20	ab97de20	1ed0	17	FWPIMX
0040	000f	0004	000f	0002	blk	0200	ab7f5000	ab99ac20	ab97ec20	1ed0	17	FWPIMX
0042	000f	0004	000f	0003	blk	0200	ab7f9000	ab99ac20	ab97f020	1ed0	17	FWPIMX
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pTDA	pTCB	Disp	SG	Name
0038	000e	0004	000e	0001	blk	0200	ab7e5000	ab99a020	ab97dc20	1ed0	16	DINFO
003f	000e	0004	000e	0002	blk	0500	ab7f3000	ab99a020	ab97ea20	1f00	16	DINFO
0037	000d	0004	000d	0001	blk	0200	ab7e3000	ab999a20	ab97da20	1ed0	15	MRFILE32
003e	000d	0004	000d	0002	blk	0200	ab7f1000	ab999a20	ab97e820		15	MRFILE32
0033	000c	0004	000c	0001	blk	0200	ab7db000	ab998e20	ab97d220	1ed0	13	PULSE
*003b	000c	0004	000c	0002	run	0100	ab7eb000	ab998e20	ab97e220	1f28	13	PULSE
003c	000c	0004	000c	0003	blk	081f	ab7ed000	ab998e20	ab97e420	1f00	13	PULSE
0026	0008	0004	0008	0001	blk	0500	ab7c1000	ab999420	ab97b820	1ed0	12	PMSHL32
002d	0008	0004	0008	0002	blk	0200	ab7cf000	ab999420	ab97c620	1edc	12	PMSHL32
002e	0008	0004	0008	0003	blk	0200	ab7d1000	ab999420	ab97c820		12	PMSHL32
002f	0008	0004	0008	0004	blk	0200	ab7d3000	ab999420	ab97ca20	1ed0	12	PMSHL32
0028	0008	0004	0008	0005	blk	0200	ab7c5000	ab999420	ab97bc20		12	PMSHL32
0025	0008	0004	0008	0006	blk	0200	ab7bf000	ab999420	ab97b620	1edc	12	PMSHL32
002c	0008	0004	0008	0007	blk	0200	ab7cd000	ab999420	ab97c420	1ed0	12	PMSHL32
0031	0008	0004	0008	0008	blk	0500	ab7d7000	ab999420	ab97ce20	1edc	12	PMSHL32
0032	0008	0004	0008	0009	blk	0200	ab7d9000	ab999420	ab97d020	1edc	12	PMSHL32
0034	0008	0004	0008	000b	blk	0500	ab7dd000	ab999420	ab97d420		12	PMSHL32
0035	0008	0004	0008	000c	blk	0200	ab7df000	ab999420	ab97d620	1eac	12	PMSHL32
0036	0008	0004	0008	000d	blk	0500	ab7e1000	ab999420	ab97d820	1eb8	12	PMSHL32
0044	0008	0004	0008	000e	blk	0200	ab7fd000	ab999420	ab97f420	1ed0	12	PMSHL32
0023	0006	0004	0006	0001	blk	0200	ab7bb000	ab998820	ab97b220		10	PMSPPOOL
0027	0006	0004	0006	0002	blk	0500	ab7c3000	ab998820	ab97ba20		10	PMSPPOOL
0029	0006	0004	0006	0003	blk	0200	ab7c7000	ab998820	ab97be20		10	PMSPPOOL
002a	0006	0004	0006	0004	blk	0500	ab7c9000	ab998820	ab97c020		10	PMSPPOOL
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pTDA	pTCB	Disp	SG	Name
002b	0006	0004	0006	0005	blk	0500	ab7cb000	ab998820	ab97c220		10	PMSPPOOL
000e	0005	0004	0005	0001	blk	0800	ab791000	ab998220	ab978820		00	HARDERR
0013	0005	0004	0005	0002	blk	0800	ab79b000	ab998220	ab979220		00	HARDERR
0014	0005	0004	0005	0003	blk	0800	ab79d000	ab998220	ab979420		00	HARDERR
0049	0018	0017	0018	0001	blk	0200	ab807000	ab99ca20	ab97fe20	1cd4	04	FROMAGE
004d#	001c	001b	001c	0001	blk	0300	ab80f000	ab99e220	ab980620	1e60	1d	PAIN

>>> Pid 18 is evidently FROMAGE.

```

>>> FROMAGE has the VIN and PAIN wants it!

>>> We had better find out why FROMAGE has blocked.

# .s 49
Current slot number: 0049

# .r
eax=00000001 ebx=00020366 ecx=1bf90000 edx=00020004 esi=13fa0000 edi=13fa1052
eip=0000a0c3 esp=0000324a ebp=00023270 iopl=2 -- -- -- nv up ei ng nz ac pe cy
cs=dfdf ss=0017 ds=9fe7 es=9fe7 fs=150b gs=0000 cr2=00000000 cr3=001d9000

Invalid linear address: dfdf:0000a0c3
# ln

dfdf:0000a053 DOSCALL1 GetCharIn + 70

>>> Looking at this from the application aspect may be difficult since
>>> some of the code has been paged out (The invalid linear address
>>> message).

>>> The near symbol gives a clue. We could try unwinding the stack and
>>> hope that the stack is still paged in.

# dw %ebp
%00023270 b17e dfdf 9fe7 0000 0000 a97a dfdf 0000
%00023280 0000 0366 9fe7 a8ec a48d 1052 a399 32a6
%00023290 203c 0053 9fdf 1052 1052 32c8 32a6 0360
%000232a0 0007 4fa8 32b4 b1f0 dfdf 9fd7 0000 0005
%000232b0 bb2b dfd7 0000 0000 1044 9fd7 1052 9fd7
%000232c0 32c8 0002 0053 1bf9 32f0 0002 1cba 1bf9
%000232d0 0000 0000 0000 0000 1044 13fa 1052 13fa
%000232e0 0000 0000 0360 0002 0360 0002 0042 0008
# d

%000232f0 3334 0002 78cd 0001 0000 0000 0000 0009
%00023300 1000 0000 3330 0002 0000 0000 0360 0002
%00023310 0360 0002 0000 0002 0000 0000 0000 0000
%00023320 0000 0000 0000 0000 0000 0000 0000 0000
%00023330 7000 ab80 3388 0002 2abc 0001 0360 0002
%00023340 0000 0009 1000 0000 33f0 0002 0029 0000
%00023350 0020 0000 0001 0000 0000 0000 0000 0000
%00023360 0000 0000 3388 0002 02c5 0001 339c 0002

>>> This is going to be haphazard. Evidently the code we are currently
>>> executing is not using EBP as a stack frame pointer. All we can do
>>> is scan through the stack looking for a likely stack frame or a
>>> return address to user code.

>>> %232f0 looks like a candidate. Let's unassemble the return address
>>> to see if it makes sense.

# u %178cd-10

%000178bd 03ca add ecx,edx
%000178bf 51 push ecx
%000178c0 8b5d08 mov ebx,dword ptr [ebp+08]
%000178c3 ff7320 push dword ptr [ebx+20]
%000178c6 b004 mov al,04
%000178c8 e83ba3f71b call %1bf91c08
%000178cd 83c410 add esp,+10
%000178d0 8bc8 mov ecx,eax
%000178d2 0bc0 or eax,eax
%000178d4 741b jz %000178f1
%000178d6 824b0802 or byte ptr [ebx+08],02
%000178da c705e81302003c000000 mov dword ptr [000213e8],0000003c
# ln %1bf91c08

%1bf91c08 DOSCALL1 DOS32READ

>>> So far so good. We need to see if this is consistent with the
>>> BlockId, which is the most up-to-date status indicator for this
>>> process.

# .pb#
Slot Sta BlockID Name Type Addr Symbol
0049# blk 05100604 FROMAGE

```

```
# .m 0510:0604

*har      par      cpgr      va      flg next prev link hash hob      hal
0003 %feaeef04c 00000400 %fe6ef000 001 0002 0023 0000 0000 0003 0000      =0000
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0003 0003 fec5 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
```

```
>>> This blockid points to data within the kernel resident read/write
>>> heap. Heap blocks have headers that tell us more about the user of
>>> the data. The data portion of a heap block is usually mapped to a
>>> GDT selector. In this example, selector 510. 510:0 should
>>> be the address of the beginning of the data and therefore point
>>> just after the end of the header. We look at the data before
>>> 510:0 to see the heap block header.
```

```
# dg 510
0510 Data Bas=fe6f3000 Lim=00000b2a DPL=0 P RW A
# dd % fe6f3000-10
```

```
%fe6f2ff0 ffa4000c feaeef28 0002036f 00000b39
%fe6f3000 05000000 0000c981 424b2d29 20202444
%fe6f3010 05182020 00000510 00020000 00000000
%fe6f3020 a8030ec8 424b0170 19002444 00000000
%fe6f3030 00000000 00000000 00000000 00000000
%fe6f3040 00000000 00061400 001b0014 0a000003
%fe6f3050 00028101 81010500 0a000001 0003c747
%fe6f3060 00000000 00000000 00000000 00000000
```

```
>>> For resident heaps the header is a double-word. This one begins
>>> at %fe6f2ffc. The low 2 bits are flags, the remainder is the
>>> length of the heap block in double-words.
```

```
>>> If the flag bit 0 is 1 then this is an extended heap. Which it is.
>>> We need to look at the header extension at the end of the block.
```

```
>>> The length of the block in bytes is b38 (by mentaly AND-ing b39
>>> and 0xffc)
```

```
# dd % fe6f3000-4+b38-10
```

```
%fe6f3b24 ffff0000 0000ffff ff530510 ff77bd64
%fe6f3b34 ffc2001c 00000008 00000000 00000000
%fe6f3b44 ab240001 4d5000a6 00005854 ff9e0014
%fe6f3b54 001b0083 fe6f3b80 fe701da0 fe8777b0
%fe6f3b64 ffa4000c fe6f3b74 000102e9 ffa4000c
%fe6f3b74 fe7ea73c 0001071f ff9e0014 001b005d
%fe6f3b84 fe83baa8 fe876eac fe877654 ffc2001c
%fe6f3b94 00000008 00000000 00000000 da680001
```

```
>>> The header extension is in the last 2 double-words of the heap
>>> block. The owner Id and the selector are in the first of these (at
>>> %fe6f3b2c).
```

```
# .mo ff53
```

```
ff53 dd4
```

```
>>> This tells us selector 510 was allocated by, or is part of the 4th
>>> device driver to initialise. Listing the physical device driver
>>> MTEs will find this. They are listed last initialised first.
```

```
>>> Note: frequently we find that dd16 is the owner. This refers to
>>> all device drivers from the 16th and subsequent. The first 15
>>> device drivers to initialise are assigned unique owner ids from
>>> dd1 to dd15, where the numbers are in decimal.
```

```
# .lmp
hmte=0249 pmte=%fe848df4 mflags=0008f1c9 h:\faxpro\fmd.sys
hmte=0242 pmte=%fe856f04 mflags=0008f1c9 c:\os2tools\theseus2.sys
hmte=0240 pmte=%fe848e7c mflags=0008f1c9 c:\os2\vdisk.sys
hmte=0235 pmte=%fe848f58 mflags=0008f1c9 h:\os2\apps\sysios2.sys
hmte=0234 pmte=%fe848f08 mflags=0008f1c9 h:\tcpip\bin\ifndisnl.sys
hmte=011e pmte=%fe83ff84 mflags=0008f1c9 h:\tcpip\bin\inet.sys
hmte=012b pmte=%fe83996c mflags=0008f1c9 h:\mmos2\r0stub.sys
hmte=012a pmte=%fe839da0 mflags=0000f1c1 h:\mmos2\ssmdd.sys
```



```

hmte=0123 pmte=%fe83df4c mflags=8008f1c9 c:\os2\com.sys
hmte=0121 pmte=%fe839e64 mflags=8008f1c9 h:\os2\boot\mouse.sys
hmte=0120 pmte=%fe839ef4 mflags=8008f1c9 h:\os2\boot\pointdd.sys
hmte=011d pmte=%fe83afdc mflags=8008f1c9 h:\os2\boot\os2cdrom.dmd
hmte=0111 pmte=%fe83af88 mflags=8008f1ca h:\os2\boot\pmdd.sys
hmte=0087 pmte=%fe839fdc mflags=0008f1c9 h:\os2\boot\dos.sys
hmte=0089 pmte=%fe839f80 mflags=8008f1c9 h:\os2\boot\testcfg.sys
hmte=0103 pmte=%fe71095c mflags=0008f1c9 i:\brew\os20memu.sys
hmte=00e2 pmte=%fe6fefc4 mflags=8008e1c9 h:\os2\scsi.dmd
hmte=00e1 pmte=%fe6fdf64 mflags=8008e1c9 h:\os2\dasd.dmd
hmte=00de pmte=%fe6f6fb0 mflags=0008e1c9 h:\xdfloppy.flt
hmte=00a9 pmte=%fe6f5fb0 mflags=8008e1c9 h:\fd16-700.add
hmte=00a7 pmte=%fe6f4f3c mflags=8008e1c9 h:\ibmls506.add
hmte=00a5 pmte=%fe6f3db4 mflags=8008e1c9 h:\ibmlflpy.add
hmte=009c pmte=%feaeaea0 mflags=8008e1c9 h:\print01.sys
hmte=009b pmte=%fe6f1fb8 mflags=8008e1c9 h:\ibmkbd.sys
hmte=009a pmte=%feaeaed8 mflags=8008e1c9 h:\kdbase.sys
hmte=0099 pmte=%feaeef34 mflags=0008e1c9 h:\screen01.sys
hmte=0098 pmte=%feaeefc0 mflags=8008e1c9 h:\clock01.sys
hmte=0096 pmte=%fe6f1fdc mflags=0008e1c9 h:\resource.sys

```

>>> Counting backwards, the 4th device driver is kdbase.sys.

>>> We dump its object table.

```

# .lmo 9a
hmte=009a pmte=%feaeaed8 mflags=8008e1c9 h:\kdbase.sys
seg sect psiz vsiz hob sel flags
0001 0001 158c 170e 0000 0510 8c41 data prel
0002 000c 3270 3270 0000 0518 8d60 code shr prel rel
0003 0026 1987 1988 0000 0520 8d60 code shr prel rel
0004 0033 0743 0744 0000 0528 8d60 code shr prel rel

```

>>> Selector 510 is indeed the first data selector of kdbase.sys and  
>>> will contain the device driver header at offset +0

```

# .d dev 510:0
    DevNext: 0500:0000
    DevAttr: c981
    DevStrat: 0000
    DevInt: 2d29
    DevName: KBD$
    DevProtCS: 0518
    DevProtDS: 0510
    DevRealCS: 0000
    DevRealDS: 0000

```

>>> We conclude that FROMAGE is waiting for the device driver to  
>>> respond to the DosRead - i.e. A keyboard interrupt

## Exploring Memory Management

This section gives a basic overview of memory management, and shows how to answer the following questions:

1. [Who owns and who allocated virtual memory?](#)
2. [How to correlate named memory with its address and \*vice versa\*](#)
3. [How memory aliasing works.](#)

If the reader is unfamiliar with this subject then these sections should be read in order.

## Who Owns Virtual Memory and Who Allocated it?

In this section we take a look at the primary system structures used in memory management and how they are located using the Dump Formatter and Kernel Debugger. These structures are:

[The memory arena record \(VMAR\)](#)

[The memory arena header record \(VMAH\)](#)

[The memory object record \(VMOB\)](#)

[The memory context record \(VMCO\)](#)

The examples worked in this section illustrate:

how to find all memory allocations made by a given process and what executable made the allocation.

how to determine ownership of non-system memory.

the use of memory objects, pseudo-objects and system objects.

Memory allocations have many attributes, included among which are:

data or content

location or address

size

ownership

requestor

The composite set of attributes associated with a memory allocation is referred to as a memory object. OS/2's virtual memory manager tracks memory objects using arena, object and context records.

We start by looking at the arena record, which is used to record virtual address assignments to memory objects.

The entire system address space of 4 gigabytes is partitioned into three types of memory arena:

#### System Arena

This is the range of virtual addresses where system information and ring0 code executes. Typically device drivers, file system drivers and the OS/2 Kernel executes and uses data assigned to the System Arena. There is just one instance of the System Arena. It is assigned the virtual address range from 512 Mb to 4 Gb.

#### Shared Arena

This is the range of virtual addresses assigned to shared objects. Shared data objects come in two varieties:

##### Global data

Such objects exist as unique entities. Their address range and data content are common to all accessing processes. This is achieved by using common page tables in all processes.

##### Instance data

Such objects share the same address range, but exist as distinct data instances in each accessing process. Page table entries for instance data are specific to each process.

Code objects from DLL modules are also consigned to the shared arena.

In general processes are not given automatic access to instance or global data. Access is granted either implicitly by the system loader because of calls to other DLLs or explicitly by use of the DosGiveXxxx and DosGetXxxx set of APIs.

There is just one shared arena, which reserves initially virtual memory addresses from 304Mb to 512Mb. This may be expanded by lowering the lower boundary. The current address range assigned to the shared arena is managed by a special arena record called the boundary sentinel arena record.

#### Private Arena

This is the range of virtual addresses used to map objects that are unique to each process. A private arena therefore exists for each process. In general the page tables of each private arena will map to unique real storage frames. An exception to this is with code objects. Since code segments are always read-only then if more than one process is running the same executable module their page tables will map to a common set of real storage frames for the code segments of the executable module.

Private arenas are assigned an initial address range from 64k to 64M. This may be expanded upwards as more memory is allocated. The current size of a private arena is tracked by a special arena record called the sentinel arena record.

The private arena upper boundary and shared arena lower boundary may grow towards each other but not overlap.

These worked examples now follow:

[Exploring arena records](#)

[Exploring object records](#)

[Finding who owns memory](#)

-----

## Exploring Arena Records

The following example illustrates the use of arena records:

```
>>> We start by asking the question: what ranges of addresses are
>>> currently allocated in the private arena of the process that's
>>> running the IPFC compiler.
```

```
>>> List all processes to find the one of interest
```

```
# .p
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
0001  0001  0000  0000  0001  blk  0100  ffe3a000  ffe3c7d4  ffe3c61c  1e7c  00  *ager
0002  0001  0000  0000  0002  blk  0200  7b7aa000  ffe3c7d4  7b9a8020  1f3c  00  *tsd
0003  0001  0000  0000  0003  blk  0200  7b7ac000  ffe3c7d4  7b9a81d8  1f50  00  *ctxh
0004  0001  0000  0000  0004  blk  081f  7b7ae000  ffe3c7d4  7b9a8390  1f48  00  *kdb
0005  0001  0000  0000  0005  blk  0800  7b7b0000  ffe3c7d4  7b9a8548  1f20  00  *lazyw
0006  0001  0000  0000  0006  blk  0800  7b7b2000  ffe3c7d4  7b9a8700  1f3c  00  *asynchr
0009  0002  0000  0002  0001  blk  021f  7b7b8000  7b9c4020  7b9a8c28      00  LOGDAEM
0008  0003  0001  0003  0001  rdy  061f  7b7b6000  7b9c484c  7b9a8a70  1eb8  01  PMSHL32
000b  0003  0001  0003  0002  blk  0800  7b7bc000  7b9c484c  7b9a8f98      01  PMSHL32
000c  0003  0001  0003  0003  blk  0800  7b7be000  7b9c484c  7b9a9150      01  PMSHL32
000d  0003  0001  0003  0004  blk  0800  7b7c0000  7b9c484c  7b9a9308      01  PMSHL32
000e  0003  0001  0003  0005  blk  0800  7b7c2000  7b9c484c  7b9a94c0      01  PMSHL32
0007  0003  0001  0003  0006  blk  0200  7b7b4000  7b9c484c  7b9a88b8  1ecc  01  PMSHL32
0011  0003  0001  0003  0007  blk  0200  7b7c8000  7b9c484c  7b9a99e8  1ecc  01  PMSHL32
0012  0003  0001  0003  0008  blk  0200  7b7ca000  7b9c484c  7b9a9ba0      01  PMSHL32
0013  0003  0001  0003  0009  blk  0200  7b7cc000  7b9c484c  7b9a9d58      01  PMSHL32
0014  0003  0001  0003  000a  blk  0800  7b7ce000  7b9c484c  7b9a9f10      01  PMSHL32
0015  0003  0001  0003  000b  blk  0800  7b7d0000  7b9c484c  7b9aa0c8      01  PMSHL32
0016  0003  0001  0003  000c  blk  0800  7b7d2000  7b9c484c  7b9aa280      01  PMSHL32
0017  0003  0001  0003  000d  blk  0804  7b7d4000  7b9c484c  7b9aa438  1ea8  01  PMSHL32
0018  0003  0001  0003  000e  rdy  0804  7b7d6000  7b9c484c  7b9aa5f0      01  PMSHL32
0019  0003  0001  0003  000f  blk  0500  7b7d8000  7b9c484c  7b9aa7a8      01  PMSHL32
001a  0003  0001  0003  0010  rdy  0801  7b7da000  7b9c484c  7b9aa960  1bac  01  PMSHL32
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
001b  0003  0001  0003  0011  blk  0800  7b7dc000  7b9c484c  7b9aab18      01  PMSHL32
*001c# 0003  0001  0003  0012  run  0800  7b7de000  7b9c484c  7b9aacd0  1b8c  01  PMSHL32
001d  0003  0001  0003  0013  blk  0200  7b7e0000  7b9c484c  7b9aae88      01  PMSHL32
0023  0018  0003  0018  0001  rdy  061f  7b7ec000  7b9c7128  7b9ab8d8  1eb8  13  EPM
0038  0018  0003  0018  0002  blk  0200  7b816000  7b9c7128  7b9adcf0  1ecc  13  EPM
0037  0013  0003  0013  0001  blk  0200  7b814000  7b9c9a04  7b9adb38      19  IBMAVSD
0033  0012  0003  0012  0001  blk  0200  7b80c000  7b9c89ac  7b9ad458  1eb8  17  PMDRAW
0035  0012  0003  0012  0002  blk  0200  7b810000  7b9c89ac  7b9ad7c8  1eb8  17  PMDRAW
0036  0012  0003  0012  0003  blk  0200  7b812000  7b9c89ac  7b9ad980      17  PMDRAW
0034  0010  0003  0010  0001  blk  0400  7b80e000  7b9c91d8  7b9ad610  1ed4  12  CMD
002e  000d  0003  000d  0001  blk  0200  7b802000  7b9c8180  7b9acbc0  1eb8  16  PULSE
0030  000d  0003  000d  0002  rdy  0100  7b806000  7b9c8180  7b9acf30  1f28  16  PULSE
002f  000d  0003  000d  0003  rdy  081f  7b804000  7b9c8180  7b9acd78  1f00  16  PULSE
002d  000c  0003  000c  0001  blk  0200  7b800000  7b9c7954  7b9aca08  1eb8  15  DINFO
0032  000c  0003  000c  0002  rdy  061f  7b80a000  7b9c7954  7b9ad2a0  1f00  15  DINFO
002c  000b  0003  000b  0001  blk  0200  7b7fe000  7b9c58a4  7b9ac850  1eb8  14  MRFILE32
0031  000b  0003  000b  0002  blk  0200  7b808000  7b9c58a4  7b9ad0e8  1ecc  14  MRFILE32
0029  000a  0003  000a  0001  rdy  061f  7b7f8000  7b9c68fc  7b9ac328  1eb8  10  PMDIARY
001f  0006  0003  0006  0001  rdy  062f  7b7e4000  7b9c60d0  7b9ab1f8  1eb8  11  PMSHL32
0021  0006  0003  0006  0002  blk  0200  7b7e8000  7b9c60d0  7b9ab568      11  PMSHL32
```

```

0022 0006 0003 0006 0003 blk 0200 7b7ea000 7b9c60d0 7b9ab720 1eb8 11 PMSHL32
0020 0006 0003 0006 0004 blk 0200 7b7e6000 7b9c60d0 7b9ab3b0 11 PMSHL32
001e 0006 0003 0006 0005 blk 0200 7b7e2000 7b9c60d0 7b9ab040 1ecc 11 PMSHL32
0024 0006 0003 0006 0006 blk 0200 7b7ee000 7b9c60d0 7b9aba90 11 PMSHL32
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0025 0006 0003 0006 0007 blk 0200 7b7f0000 7b9c60d0 7b9abc48 11 PMSHL32
0026 0006 0003 0006 0008 blk 0200 7b7f2000 7b9c60d0 7b9abe00 11 PMSHL32
0027 0006 0003 0006 0009 blk 0200 7b7f4000 7b9c60d0 7b9abfb8 11 PMSHL32
0028 0006 0003 0006 000a blk 0200 7b7f6000 7b9c60d0 7b9ac170 11 PMSHL32
002a 0006 0003 0006 000c blk 021f 7b7fa000 7b9c60d0 7b9ac4e0 1eac 11 PMSHL32
002b 0006 0003 0006 000d blk 0200 7b7fc000 7b9c60d0 7b9ac698 1eb8 11 PMSHL32
000a 0004 0003 0004 0001 blk 0800 7b7ba000 7b9c5078 7b9a8de0 00 HARDERR
000f 0004 0003 0004 0002 blk 0800 7b7c4000 7b9c5078 7b9a9678 00 HARDERR
0010 0004 0003 0004 0003 blk 0800 7b7c6000 7b9c5078 7b9a9830 00 HARDERR
0039 0019 0010 0019 0001 rdy 061f 7b818000 7b9ca230 7b9adea8 1f0c 12 IPFC

```

```

>>> From the name printed in the right hand column we see that slot 39
>>> is the one of interest.

```

```

>>> Imbedded in each PTDA at offset +0x40 is the VMAH that heads the
>>> private arena. From the VMAH we can obtain the pointer to the
>>> sentinel area record.

```

```

>>> Dump out the VMAH for slot 39 using the pPTDA address from the
>>> .p command output...

```

```

# dd %7b9ca230+40 Ll0
%7b9ca270 7b9c7168 fff13190 feb24cae feb261bc
%7b9ca280 fe79ba54 fe87e9a0 fff03e30 00000002
%7b9ca290 00010000 00370000 00000000 00000000
%7b9ca2a0 00000003 00000000 00000041 000005db

```

```

>>> The third double word (feb24cae) is the address of the sentinel
>>> record. To format this using the .MA command we need to determin
>>> the handle for this record. Arena records are organised in a
>>> table of 0x16 byte length entries. Their handles are their
>>> corresponding table entry number. The address of the first
>>> arena record is located at symbol _parvmone...

```

```

# dd _parvmone ll
%fff13304 feblf020

```

```

>>> Arena record 1 is located at %feblf020. We wish to determine the
>>> handle for the sentinel, whose address is %feb24cae. We use the
>>> hex calculator facility of the Dump Formatter/Kernel Debugger thus..

```

```

# ? (%feb24cae-%feblf020)/16 +1
%00000436

```

```

>>> The handle we require is 436. We can now format the sentinel
>>> for slot 39 ....

```

```

# .ma 436
har par cpg va flg next prev link hash hob hal
0436 %feb24cae 00000000 %00010000 003 050a 0526 0005 0000 4000 0000 max=%04000000

```

```

>>> Note the max=%04000000 to the right indicating the current private
>>> arena maximum address is 64M - 1 and incidentally distinguishing
>>> this as a sentinel or boundary sentinel arena record.
>>> Note also that this is merely a boundary marker and not an indication
>>> of which addresses within the private arena have been allocated.

```

```

>>> Regular arena record are chained to the sentinel in a circular
>>> double linked list using the 'next' and 'prev' pointers.
>>> We can format the entire chain using .MAL (or .MAR) but we have to
>>> break in using Ctrl-C to stop the chain endlessly traversing the
>>> circular chain.

```

```

# .ma 50a
har par cpg va flg next prev link hash hob hal
050a %feb25ee6 00000030 %00010000 1d9 0509 0436 0000 0000 05e2 0000 hptda=050c
# .ma 509
har par cpg va flg next prev link hash hob hal
0509 %feb25ed0 00000010 %00040000 179 050b 050a 0000 0000 05dd 0000 hptda=050c
# .ma 50b
har par cpg va flg next prev link hash hob hal
050b %feb25efc 00000010 %00050000 169 0507 0509 0000 0000 05e9 0000 hptda=050c
# .mal 507

```

```

har      par      cpg      va      flg next prev link hash hob      hal
0507 %feb25ea4 00000010 %00060000 169 0506 050b 0000 0000 05ea 0000 hptda=050c
0506 %feb25e8e 00000010 %00070000 169 050f 0507 0000 0000 05eb 0000 hptda=050c
050f %feb25f54 00000010 %00080000 169 050c 0506 0000 0000 05ec 0000 hptda=050c
050c %feb25f12 00000010 %00090000 169 0511 050f 0000 0000 05ed 0000 hptda=050c
0511 %feb25f80 00000010 %000a0000 169 050e 050c 0000 0000 05f0 0000 hptda=050c
050e %feb25f3e 00000010 %000b0000 1c9 050d 0511 01c7 0000 05ee 0016 hptda=050c
050d %feb25f28 00000010 %000c0000 169 0512 050e 0000 0000 05f1 0000 hptda=050c
0512 %feb25f96 00000010 %000d0000 169 0513 050d 0000 0000 05f2 0000 hptda=050c
0513 %feb25fac 00000010 %000e0000 169 0514 0512 0000 0000 05f3 0000 hptda=050c
0514 %feb25fc2 00000010 %000f0000 169 0515 0513 0000 0000 05f4 0000 hptda=050c
0515 %feb25fd8 00000010 %00100000 169 0516 0514 0000 0000 05f5 0000 hptda=050c
0516 %feb25fee 00000010 %00110000 169 0517 0515 0000 0000 05f6 0000 hptda=050c
0517 %feb26004 00000010 %00120000 169 0519 0516 0000 0000 05f7 0000 hptda=050c
0519 %feb26030 00000010 %00130000 169 0518 0517 0000 0000 05f9 0000 hptda=050c
0518 %feb2601a 00000010 %00140000 169 051a 0519 0000 0000 05f8 0000 hptda=050c
051a %feb26046 00000010 %00150000 169 051b 0518 0000 0000 05fa 0000 hptda=050c
051b %feb2605c 00000010 %00160000 169 051c 051a 0000 0000 05fb 0000 hptda=050c
051c %feb26072 00000010 %00170000 169 051d 051b 0000 0000 05fc 0000 hptda=050c
051d %feb26088 00000010 %00180000 169 051e 051c 0000 0000 05fd 0000 hptda=050c
051e %feb2609e 00000010 %00190000 169 0521 051d 0000 0000 05fe 0000 hptda=050c
0521 %feb260e0 00000010 %001a0000 169 0520 051e 0000 0000 0601 0000 hptda=050c
0520 %feb260ca 00000010 %001b0000 169 051f 0521 0000 0000 0600 0000 hptda=050c
051f %feb260b4 000000f0 %001c0000 169 0523 0520 0000 0000 05ff 0000 hptda=050c
har      par      cpg      va      flg next prev link hash hob      hal
0523 %feb2610c 00000010 %002b0000 169 0527 051f 0000 0000 0603 0000 hptda=050c
0527 %feb26164 00000010 %002c0000 169 0522 0523 0000 0000 0607 0000 hptda=050c
0522 %feb260f6 00000020 %002d0000 169 0525 0527 0000 0000 0602 0000 hptda=050c
0525 %feb26138 00000010 %002f0000 169 0524 0522 0000 0000 0605 0000 hptda=050c
0524 %feb26122 00000010 %00300000 169 052a 0525 0000 0000 0604 0000 hptda=050c
052a %feb261a6 00000010 %00310000 169 052d 0524 0000 0000 060a 0000 hptda=050c
052d %feb261e8 00000010 %00320000 169 0529 052a 0000 0000 060d 0000 hptda=050c
0529 %feb26190 00000010 %00330000 169 052b 052d 0000 0000 0609 0000 hptda=050c
052b %feb261bc 00000020 %00340000 169 0526 0529 0000 0000 060b 0000 hptda=050c
0526 %feb2614e 00000010 %00360000 169 0436 052b 0000 0000 0606 0000 hptda=050c
0436 %feb24cae 00000000 %00010000 003 050a 0526 0005 0000 4000 0000 max=%04000000
050a %feb25ee6 00000030 %00010000 1d9 0509 0436 0000 0000 05e2 0000 hptda=050c
0509 %feb25ed0 00000010 %00040000 179 050b 050a 0000 0000 05dd 0000 hptda=050c

```

#

```

>>> Each regular private arena record is distinguished by the appearance
>>> hptda=nnn to the right of each line. This is the handle of the PTDA
>>> of the process to which the arena record belongs. Each of the hptda
>>> values is 50c indicating each of regular arena records above belongs
>>> to the same process. More on the hptda later.

```

```

>>> Each regular arena represents the address range reserved for a
>>> memory object. cpg is the size reservation in pages, but note
>>> that this is only an address space reservation, not necessarily what
>>> is currently committed. Most objects reserve 0x10
>>> pages or 64K, which corresponds to the maximum 16-bit segment size.

```

```

>>> va shows the start address of each memory object.
>>> By examining va and cpg we can see that the minimum and maximum
>>> addresses allocated in the private arena of slot 39 is %10000 and
>>> %36ffff (=%360000 + 0x10 pages -1). We can also see that this
>>> allocation is contiguous and therefore the total allocated private
>>> arena virtual address space is 0x360000 bytes or 3.375M

```

```

>>> The VMAH records the minimum and maximum +1 allocated addresses
>>> at +0x20 and +0x24, but the allocation might be sparse so the VMAH
>>> does not indicate directly the total memory in use.

```

```

>>> We now move onto the shared arena.

```

```

>>> The link field of each sentinel points to the boundary sentinel

```

```

# .ma 436
har      par      cpg      va      flg next prev link hash hob      hal
0436 %feb24cae 00000000 %00010000 003 050a 0526 0005 0000 4000 0000 max=%04000000
# .ma 5
har      par      cpg      va      flg next prev link hash hob      hal
0005 %feblf078 00011a20 %04000000 007 0508 0075 0000 0000 fff0 0000 max=%1fff0000

```

```

>>> Once again each regular arena record in the shared arena is linked
>>> in a circular double linked list. This time we enter the chain from
>>> the boundary sentinel next and prev fields.

```

```
# .mal 508
har    par      cpq      va      flg next prev link hash hob   hal
0508 %feb25eba 00000010 %15a20000 369 0437 0005 0000 0000 05df 0000 hco=008a8
0437 %feb24cc4 00000010 %15a40000 369 0438 0508 0000 0000 050d 0000 hco=00248
0438 %feb24cda 00000010 %15a50000 369 0444 0437 0000 0000 050e 0000 hco=0076e
0444 %feb24de2 00000020 %15a60000 3d9 0441 0438 0000 0000 0518 0000 hco=007aa
0441 %feb24da0 00000010 %15a80000 3d9 043b 0444 0000 0000 051a 0000 hco=007a9
043b %feb24d1c 00000010 %15a90000 3d9 043a 0441 0000 0000 0517 0000 hco=002b7
043a %feb24d06 00000010 %15aa0000 3d9 0443 043b 0000 0000 0511 0000 hco=007a8
0443 %feb24dcc 00000010 %15ab0000 179 0439 043a 0000 0000 0519 0000 =0000
0439 %feb24cf0 00000010 %15ac0000 369 0433 0443 0000 0000 050f 0000 hco=00763
0433 %feb24c6c 00000010 %15ad0000 369 0432 0439 0000 0000 0509 0000 hco=00777
0432 %feb24c56 00000010 %15ae0000 369 041e 0433 0000 0000 0508 0000 hco=00776
041e %feb24a9e 00000030 %15af0000 369 041c 0432 0000 0000 04f4 0000 hco=007d8
041c %feb24a72 00000010 %15b20000 369 03ee 041e 0000 0000 04d1 0000 hco=0075c
03ee %feb2467e 00000010 %15b30000 349 03eb 041c 0000 0000 04c1 0000 hco=001f6

.
.
.
.
.

0169 %feb20f10 00000010 %1acb0000 179 0168 016a 0000 0000 05e6 0000 =0000
0168 %feb20efa 00000020 %1acc0000 379 0077 0169 0000 0000 01af 0000 hco=007c0
0077 %feb1fa44 00000010 %1bfe0000 349 0075 0168 0000 0000 0077 0000 hco=007a7
0075 %feb1fa18 00000010 %1bff0000 349 0005 0077 0000 0000 0075 0000 hco=007b8
0005 %feb1f078 00011a20 %04000000 007 0508 0075 0000 0000 fff0 0000 max=%1fff0000
0508 %feb25eba 00000010 %15a20000 369 0437 0005 0000 0000 05df 0000 hco=008a8
0437 %feb24cc4 00000010 %15a40000 369 0438 0508 0000 0000 050d 0000 hco=00248
```

```
>>> There are two types of regular arena record that appear in the
>>> Shared arena. These are distinguished by the right-hand column:
>>> hco=nnnnn
>>> =0000
>>> The first type is global shared data. The hco is the context record
>>> handle, which will be discussed later.
>>> The second type represents instance data. Both of these will be
>>> looked at in more detail in the next section.
```

```
>>> Finally we look at the system arena. The sentinel for the system
>>> arena is har=4. Once again each regular arena record is linked
>>> in a circular double-linked list.
```

```
# .ma 4
har    par      cpq      va      flg next prev link hash hob   hal
0004 %feb1f062 00000000 %60000000 003 0504 0016 0000 0000 ffc0 0000 max=%fffc0000
# .mal 504
har    par      cpq      va      flg next prev link hash hob   hal
0504 %feb25e62 00000010 %79eb7000 121 03d2 0004 0000 0081 05e1 0000 =0000
03d2 %feb24416 00000010 %79ec7000 121 0363 0504 0000 0080 049e 0000 =0000
0363 %feb23a8c 00000010 %79ed7000 121 0374 03d2 0000 007f 0434 0000 =0000
0374 %feb23c02 00000010 %79ee7000 121 02e4 0363 0000 0095 0422 0000 =0000
02e4 %feb22fa2 00000010 %79ef7000 121 02db 0374 0000 00df 0382 0000 =0000
02db %feb22edc 00000010 %79f07000 121 02cc 02e4 0000 007c 036e 0000 =0000
02cc %feb22d92 00000010 %79f17000 121 0405 02db 0000 007b 0350 0000 =0000

.
.
.
.

0012 %feb1f196 00000016 %ffefe000 001 0013 0011 0000 0000 0013 0000 =0000
0013 %feb1flac 00000010 %fff14000 009 0014 0012 0000 0000 0014 0000 sel=0150
0014 %feb1flc2 0000000a %fff24000 009 0015 0013 0000 0000 0015 0000 sel=0158
0015 %feb1fld8 00000010 %fff2e000 009 0016 0014 0000 0000 0016 0000 sel=0160
0016 %feb1fleee 00000082 %fff3e000 001 0004 0015 0000 0000 0017 0000 =0000
0004 %feb1f062 00000000 %60000000 003 0504 0016 0000 0000 ffc0 0000 max=%fffc0000
0504 %feb25e62 00000010 %79eb7000 121 03d2 0004 0000 0081 05e1 0000 =0000
03d2 %feb24416 00000010 %79ec7000 121 0363 0504 0000 0080 049e 0000 =0000
```

```
>>> Two types of regular record appear, distinguished by the right-hand
>>> column:
>>> =0000
>>> sel=nnnn
>>> The first of these indicates heap data. The second GDT selector
```

```
>>> assigned data. Device driver and IFS code and data objects will
>>> appear among these.
```

## Exploring Object Records

We now explore the [memory object record \(VMOB\)](#) and the [.MO command](#).

```
>>> There are two types of object managed by object records:
>>>     pseudo-objects
>>>     non-pseudo-object
```

```
>>> Non-pseudo-objects have an associated arena record. These are by
>>> far the most common type of memory object. They include code and
>>> data segments of application and system code. However, there is
>>> a draw-back in that arena records deal with page size quantities.
>>> For certain system control blocks it is useful to have distinct
>>> objects associated with each instance of them. But these objects
>>> are generally very much smaller than a page. To overcome these
>>> difficulties such objects are sub-allocated from the system
>>> heap and given an object type of pseudo-object. They have
>>> no associated arena record.
```

```
>>> We list a few pseudo-objects. Note the 'p' parameter of .mo to
>>> do this.
```

```
# .mop
hob      va      flgs own  hmte  sown,cnt lt st xf
0004 %fff13238 8000 ffe1 0000 0000 00 00 00 00 vmah
0005 %fff13190 8000 ffe1 0000 0000 00 00 00 00 vmah
0006 %fff0a891 8000 ffa6 0000 0000 00 00 00 00 mte
0072 %ffe3c7d4 8000 ffc3 0000 0000 00 00 00 00 ptda 0001 doscalls.dll
007a %fff0b3fa 8000 ffa6 0000 0000 00 00 00 00 mte *sysinit
007b %fff0b26b 8000 ffa6 0000 0000 00 00 00 00 mte mvdm.dll
007d %fe720f60 8000 ffa6 0000 0000 00 00 00 00 mte fshelper.dll
0086 %fe861ee0 8000 ffa6 0000 0000 00 00 00 00 mte a:mini_fsd.fsd
0087 %fe861f30 8000 ffa6 0000 0000 00 00 00 00 mte c:pmdd.sys
0088 %fe861f58 8000 ffa6 0000 0000 00 00 00 00 mte c:dos.sys
008a %fe860f9c 8000 ffa6 0000 0000 00 00 00 00 mte c:testcfg.sys
0091 %7b9c484c 8000 ffc3 ff79 0000 00 00 00 00 ptda 0003 c:pmshapim.dll
0096 %fe721fb8 8000 ffa6 0000 0000 00 00 00 00 mte c:pmshell.exe
0097 %fe721ffc 8000 ffa6 0000 0000 00 00 00 00 mte c:clock01.sys
0098 %fe721eb0 8000 ffa6 0000 0000 00 00 00 00 mte c:screen01.sys
0099 %fe7246bc 8000 ffa6 0000 0000 00 00 00 00 mte c:kbd01.sys
009f %fe724f84 8000 ffa6 0000 0000 00 00 00 00 mte c:print01.sys
00a1 %fe725f88 8000 ffa6 0000 0000 00 00 00 00 mte c:ibm1flpy.add
c:ibm1s506.add
```

```
#
```

```
>>> Pseudo-objects apply to four types of control block but in general
>>> we will only be concerned with the PTDA and the MTE.
>>> The pseudo-object record is distinguished by the presence of the
>>> 'va' field. The control block name is shown in column 11.
```

```
>>> |
>>> .....|
>>> |      |
>>> V      V
```

```
# .mo 91
hob      va      flgs own  hmte  sown,cnt lt st xf
0091 %7b9c484c 8000 ffc3 ff79 0000 00 00 00 00 ptda 0003 c:pmshell.exe
```

```
>>> The 'va' field gives the address of the object itself. In this
>>> it's a PTDA address. We can find the thread slots which correspond
>>> to this PTDA either by using .P and looking for a match in the
>>> pPTDA field or directly:
```

```
# dw %7b9c484c+pid-ptda_start 11
%7b9c5042 0003
>>> This is the pid. Note .mo 91 extracts this for us - the pid appears
```

```
>>> after 'ptda'

# dd %7b9c484c+ptda_pTCBHead-ptda_start 11
%7b9c486c 7b9a8a70
>>> This is the head of the TCB tree for pid 3.

# dw 7b9a8a70 12
%7b9a8a70 0001 0008
>>> Words 0 and 1 of the TCB contain the thread ordinal and its slot
>>> number. This is tid 1 in slot 8.

# .p8
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0008 0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01 PMSHL32

>>> The PTDA for slot 8 has pid 3, is at %7b9c484c which is hob 91.
>>> The handle of the PTDA (hptda) is defined to be the hob of the
>>> object it occupies. Thus this ptda is also identified by hptda=91

>>> The other most frequently encountered pseudo-object is the mte.

hob va flgs own hmte sown,cnt lt st xf
0193 %fe722dec 8000 ffa6 0000 0000 00 00 00 00 mte c:pmsHELL.exe

>>> The MTE represents a loaded module. In this case the MTE control
>>> block is located at %fe722dec and is assigned the mte handle of its
>>> hob. In this case the MTE at %fe722dec is also referred to as
>>> hmte=193. The .LM command will respond to either hmte or MTE address
>>> and format the MTE for us ...
# .lm 193
hmte=0193 pmte=%fe722dec mflags=84903150 c:\os2\pmsHELL.exe

>>> .MO extracts the module name from the MTE and displays this to the
>>> right of 'mte'

>>> In each object record is the 'own' and 'hmte' fields. These are used
>>> to attribute ownership and associate a module or part of the system
>>> that was involved with the allocation request. In many cases
>>> these fields contain hobs of related objects. In some cases
>>> attribution needs to be made to a system resource. For this a
>>> number of generic system object Ids have been defined. They all
>>> are greater than 0xff00. .MO will translate system object Ids into
>>> a more meaningful text string. For example: in hob 193 the
>>> owner id ffa6
# .mo ffa6
ffa6 ldrmte

>>> Similarly in hob 91 the owner is ffcb
# .mo ffcb
ffcb ptda

>>> Both of these give an indication of the type of system object.
>>> In the first case a load MTE, in the second a PTDA. The 'own' and
>>> 'hmte' interpretation is used to form the description that appears
>>> to the right of each .MO line.

>>> We now turn our attention to non-pseudo objects or normal
>>> memory objects:
```

```
.mon
hob har hobnxt flgs own hmte sown,cnt lt st xf
0001 0001 fec8 0000 fff1 0000 0000 00 00 00 00 vmob
0002 0002 fec8 0000 ffe3 0000 0000 00 00 00 00 vmar
0003 0003 fec8 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
0007 0006 0000 0000 ff6d 0000 0000 00 00 00 00 doshlp
0008 0007 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
0009 0008 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000a 0009 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000b 000a 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000c 000b 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000d 000c 0000 0325 ffba 0000 0000 00 00 00 00 lock
000e 000d 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
000f 000e 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
0010 0087 0000 402c 0091 019f 0000 00 00 00 00 priv 0003 c:pmsHELL.exe
0011 0010 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
0012 0011 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
0013 0012 0000 0000 ffaa 0006 0000 00 00 00 00 os2krnl
```



```

.
.
.
.
009d 0096 0000 0225 ff8c 0000 0000 00 00 00 00 perfvie
009e 0097 0000 0524 ff88 ff54 0000 00 00 00 00 ptogdt dd5
00a0 0098 0000 0524 ff88 ff56 0000 00 00 00 00 ptogdt dd7
00a3 0099 0000 0524 ff88 ff56 0000 00 00 00 00 ptogdt dd7
00a4 009a 0000 0524 ff88 ff56 0000 00 00 00 00 ptogdt dd7
00a5 009b 0000 0524 ff88 ff56 0000 00 00 00 00 ptogdt dd7
00a6 009c 0000 0524 ff88 ff56 0000 00 00 00 00 ptogdt dd7
.
.
.
00e1 00d5 0000 0324 ff93 0000 0000 00 00 00 00 fsbuf
00e2 00d6 0000 482c fff7 019f 0000 00 00 00 04 giveget
00e3 00d7 0000 0124 ff8f 0000 0000 00 00 00 00 resource
00e4 01c8 0000 0824 0131 0131 0000 00 00 00 00 shared c:display.dll
hob har hobnxt flgs own hmte sown,cnt lt st xf
00e6 00d9 0000 082c 00e5 00e5 0000 00 00 00 00 shared c:doscalls.dll
00e7 00da 0000 0838 00e5 00e5 0000 00 00 00 00 shared c:doscalls.dll
00e8 00db 0000 0838 00e5 00e5 0000 00 00 00 00 shared c:doscalls.dll
00e9 00dc 0000 0838 00e5 00e5 0000 00 00 00 00 shared c:doscalls.dll
00ea 00dd 0000 0838 00e5 00e5 0000 00 00 00 00 shared c:doscalls.dll
00eb 00de 0000 0830 00e5 00e5 0000 00 00 00 00 shared c:doscalls.dll
00ec 00f1 0000 422c 0091 01b0 0000 00 00 00 00 priv 0003 c:pmshe
00ed 010d 0000 402c 0091 01b0 0000 00 00 00 00 priv 0003 c:pmshe
00ee 00e1 0000 082c 00f1 00f1 0000 00 00 00 00 shared c:os2char.dll
00f2 01c9 0000 0824 0131 0131 0000 00 00 00 00 shared c:display.dll
00f3 00e3 0000 0838 00f1 00f1 0000 00 00 00 00 shared c:os2char.dll
00f4 00e4 0000 482c fff7 00f1 0000 00 00 00 00 giveget
00f6 00e5 0000 082c 00fb 00fb 0000 00 00 00 00 shared c:sesmgr.dll
00fc 00e6 0000 082c 00fb 00fb 0000 00 00 00 00 shared c:sesmgr.dll
00fd 00e7 0000 0838 00fb 00fb 0000 00 00 00 00 shared c:sesmgr.dll
00fe 00e8 0000 0838 00fb 00fb 0000 00 00 00 00 shared c:sesmgr.dll
00ff 00e9 0000 082c 0100 0100 0000 00 00 00 00 shared c:quecalls.dll
0101 00ea 0000 082c 0100 0100 0000 00 00 00 00 shared c:quecalls.dll
0102 00eb 0000 082c 0100 0100 0000 00 00 00 00 shared c:quecalls.dll
0103 00ec 0000 0838 0100 0100 0000 00 00 00 00 shared c:quecalls.dll
0104 00ed 0000 0838 0100 0100 0000 00 00 00 00 shared c:quecalls.dll
0105 00ee 0000 0838 0100 0100 0000 00 00 00 00 shared c:quecalls.dll
0106 01cd 0000 1024 0091 0131 0000 00 00 00 00 priv 0003 c:pmshe
hob har hobnxt flgs own hmte sown,cnt lt st xf
0107 00f0 0000 0124 ffc4 0000 0000 00 00 00 00 smdfh
0108 010f 0000 4a2c fff5 01b0 0000 00 00 00 00 give
0109 0085 0000 0524 ff88 ff5b 0000 00 00 00 00 ptogdt dd12
010b 0084 0000 0524 ff88 ff5b 0000 00 00 00 00 ptogdt dd12
010c 0083 0000 0524 ff88 ff5b 0000 00 00 00 00 ptogdt dd12
010d 00f2 0000 0524 ff88 ff5b 0000 00 00 00 00 ptogdt dd12
.
.
.

```

#

```

>>> Many of these objects have a system ID owners but those of current
>>> interest are objects allocated within the shared and private arenas
>>> by application programs.

```

```

>>> Private arena private data:
>>> -----

```

```

>>> We start by examining hob 10 in more detail.

```

```

>>> We list the object and its associated arena record using the 'c'
>>> parameter of .MO
# .moc 10

```

```

*har    par    cp    va    flg next prev link hash hob    hal
0087 %feblfba4 00000010 %00070000 169 01a0 019a 0000 0000 0010 0000 hptda=0091
hob    har hobnxt flgs own hmte sown,cnt lt st xf
0010 0087 0000 402c 0091 019f 0000 00 00 00 00 priv 0003 c:pmshe

```

```

>>> We can tell from its location (%70000) that this is a private
>>> arena address in process hptda=91. The 'own' field of the object
>>> record is also hob 91 which again implies an object owned by the
>>> process. That means the object is either a dynamic allocation or

```

>>> a non-shared segment of the EXE load module, for example it's  
>>> stack segment.

```
# .mo 91
hob      va      flgs own  hmte  sown,cnt lt st xf
0091     %7b9c484c 8000 ffcf ff79 0000 00 00 00 00 ptda 0003 c:\pmsHELL.exe
```

>>> This tells us the owner is a PTDA (that is, a process private arena)  
>>> and the PID is 3, which is executing PMSHELL.EXE  
>>> Note: the PID and executable have been extracted from hob 91 and  
>>> displayed in the description area of hob 10.

>>> Now look at the hmte for hob 10.

```
# .mo 19f
hob      va      flgs own  hmte  sown,cnt lt st xf
019f     %fe8629e0 8000 ffa6 0000 0000 00 00 00 00 mte      c:\pmwin.dll
```

>>> This is the MTE for pmwin.dll.

>>> The 'own' and 'hmte' of hob 10 tell us that hob 10 was allocated in  
>>> the private arena of process PID 3 by pmwin.dll as a result of a  
>>> direct or indirect call to pmwin from pmsHELL.

>>> The flags in hob 10 can give us more information on the  
>>> characteristics of hob 10

```
>>> 4      0      2      c
      0100 0000 0010 1100
      |      |      |
      |      |      |. writeable
      |      |      |...user storage
      |      |      |.....executable
      |.....API located
```

>>> The combination writeable + executable should be interpreted as  
>>> R/W storage rather than executable storage. Looking at the page  
>>> table entry for %7000 in slot 8 (pid 3) will confirm this:

```
# .s8
Current slot number: 0008
# .p8
Slot Pid Ppid Csid Ord Sta Pri pTSD      pPTDA      pTCB      Disp SG Name
0008 0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01 PMSHL32
# dp %70000
linaddr  frame  pteframe  state res Dc Au CD WT Us rW Pn state
%00070000* 012f3  frame=0120d  0    0 D A      U W P  pageable
%00070000 0120d  frame=0120d  0    0 D A      U W P  pageable
#
```

>>> Private arena shared data:  
>>> -----

```
# .moc 192
*har      par      cpq      va      flg next prev link hash hob      hal
0249 %feb22250 00000010 %00010000 1c9 024a 0247 014e 0000 0192 0000 hptda=02a6
014e %feb20cbe 00000010 %00010000 1d9 014f 008e 0000 0000 0192 0000 hptda=0091
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0192 0249 0000 0838 0193 0193 0000 00 00 00 00 shared      c:\pmsHELL.exe
```

>>> Object 192 has two private arena records pointing to it. One  
>>> associated with hptda=2a6 and the other with hptda=91. We established  
>>> earlier that hptda=91 is pid 3 and was running pmsHELL.exe

```
# .mo 2a6
hob      va      flgs own  hmte  sown,cnt lt st xf
02a6     %7b9c60d0 8000 ffcf 0000 0000 00 00 00 00 ptda 0006 c:\pmsHELL.exe
```

>>> So hptda=2a6 refers to pid 6, which is also running pmsHELL.exe  
>>> Note the use of the 'link' field in har=249 to point to har=14e. The two  
>>> arena records are chained in this way to link all arena records that  
>>> share a private data object. The object record points to the head of  
>>> the chain.

>>> The 'own' and 'hmte' fields both point to object 193. This tells us  
>>> that object 193 is a shared segment of the load module whose  
>>> handle is 193.  
>>> This may be verified as follows...

```
# .mo 193
hob      va      flgs own  hmte  sown,cnt lt st xf
```

```
0193 %fe722dec 8000 ffa6 0000 0000 00 00 00 00 mte c:\pmsHELL.exe
```

```
# .lmo 193
hmte=0193 pmte=%fe722dec mflags=84903150 c:\os2\pmsHELL.exe
obj vsize vbase flags ipagemap cpagemap hob sel
0001 00000600 00010000 80002025 00000001 00000001 0192 000f r-x shr big
0002 0000005c 00020000 80002003 00000002 00000001 0000 0017 rw- big
0003 0000fa20 00030000 80002003 00000003 00000001 0000 001f rw- big
```

```
>>> We actually discover this is object 1 of the pmsHELL.exe load
>>> module.
```

```
>>> Examining the flags from hob 192 we see:
```

```
>>> 0      8      3      8
>>> 0000 1000 0011 1000
>>>      '      '      '
>>>      '      '      '.... User storage
>>>      '      '..... Readable
>>>      '      '..... Executable
>>>      '..... Shared
>>>
```

```
>>> This is information summarised in the description field of hob 192.
```

```
>>> Finally we take a look at the page table entries for %10000 in pid 3
>>> and 6. We should see the same real storage frame being accessed by
>>> both processes:
```

```
# .s8
Current slot number: 0008
# .p8
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0008 0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01 PMSHL32
# dp %10000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00010000* 012f3 frame=011cb 0 0 c A U r P pageable
%00010000 011cb frame=011cb 0 0 c A U r P pageable
# .s lf
Current slot number: 001f
# .p lf
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
001f# 0006 0003 0006 0001 rdy 062f 7b7e4000 7b9c60d0 7b9ab1f8 1eb8 11 PMSHL32
# dp %10000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00010000* 0093d frame=011cb 0 0 c A U r P pageable
%00010000 011cb frame=011cb 0 0 c A U r P pageable
#
```

```
>>> In both processes %00010000 translates to %011cb000
```

```
>>> Shared Arena, Global Data:
```

```
>>> -----
```

```
# .moc e6
```

```
*har par cpq va flg next prev link hash hob hal
00d9 %feb202b0 00000010 %1a060000 379 00d8 00da 0000 0000 00e6 0000 hco=0075d
hob har hobnxt flgs own hmte sown,cnt lt st xf
00e6 00d9 0000 082c 00e5 00e5 0000 00 00 00 00 shared c:\dos\call11.dll
hco=075d pco=fe6804ec hconext=0084c hptda=050c f=16 pid=0019 e:ipfc.exe
hco=084c pco=fe680997 hconext=006de hptda=04c6 f=16 pid=0018 c:epm.exe
hco=06de pco=fe680271 hconext=00660 hptda=049c f=16 pid=0013 d:ibmavsd.exe
hco=0660 pco=fe67fffb hconext=00497 hptda=0410 f=16 pid=0012 c:pmdraw.exe
hco=0497 pco=fe67f70e hconext=0036b hptda=0420 f=16 pid=0010 c:cmd.exe
hco=036b pco=fe67f132 hconext=00327 hptda=0380 f=16 pid=000d c:pulse.exe
hco=0327 pco=fe67efde hconext=001a0 hptda=036c f=16 pid=000c c:dinfo.exe
hco=01a0 pco=fe67e83b hconext=002c6 hptda=034e f=16 pid=000b c:mrfile32.exe
hco=02c6 pco=fe67edf9 hconext=0014c hptda=0317 f=16 pid=000a c:pmdiary.exe
hco=014c pco=fe67e697 hconext=000a2 hptda=02a6 f=16 pid=0006 c:pmsHELL.exe
hco=00a2 pco=fe67e345 hconext=00033 hptda=0205 f=16 pid=0004 c:harderr.exe
hco=0033 pco=fe67e11a hconext=00029 hptda=0091 f=16 pid=0003 c:pmsHELL.exe
hco=0029 pco=fe67e0e8 hconext=00000 hptda=0169 f=16 pid=0002 c:logdaem.exe
```

```
>>> We can tell immediately that this is shared arena global data from the
>>> presence of hco= in the arena record. The hco is the handle to the
>>> context record. These record the hptda of the process that is accessing
>>> shared global data. Each of the VMCOs, that's sharing the same object
>>> is chained in a single linked list from the arena record.
```

>>> The description to the right of each VMCO is derived from the hptda object.

>>> Note: the .MC command formats a VMCO. Under the Dump Formatter the VMCO chain is not run to completion so we must run the chain manually by using the hconext= field as the VMCO chain pointer.

>>> The 'own' and 'hnte' fields being equal indicate that the object is part of DOSCALL1.DLL. We can check out which object in DOSCALL1 using .lmo

```
#.lmo e5
hnte=00e5 pmte=%fe72dfac mflags=8498b594 c:\os2\dll\doscall1.dll
obj  vsize  vbase  flags  ipagemap  cpagemap  hob  sel
0001 00001354 1a010000 80009025 00000001 00000002 00eb d00e r-x shr alias iopl
0002 0000ced0 1a020000 80002025 00000003 0000000d 00ea d017 r-x shr big
0003 00001928 1a030000 80001025 00000010 00000002 00e9 d01f r-x shr alias
0004 000002ce 1a040000 80001025 00000012 00000001 00e8 d027 r-x shr alias
0005 000054f8 1a050000 8000d025 00000013 00000006 00e7 d02e r-x shr alias conf iopl
0006 00000280 1a060000 80001023 00000019 00000001 00e6 d037 rw- shr alias
0007 00001b40 1a070000 80001003 0000001a 00000002 0000 d03f rw- alias
```

>>> hob e6 is load module object 6 of doscall1.dll. Furthermore it is a read/write object. We can illustrate this by looking at the page table entries for two of the processes that are accessing hob e6.

```
# .s8
Current slot number: 0008
# .p8 Slot Pid Ppid Csid Ord Sta Pri pTSD pTDA pTCB Disp SG Name
0008# 0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01 PMSHL32
# dp %1a060000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a060000* 012fe frame=00ele 0 0 D u U W P pageable
%1a060000 00ele frame=00ele 0 0 D u U W P pageable
# .s1f
Current slot number: 001f
# .p1f
Slot Pid Ppid Csid Ord Sta Pri pTSD pTDA pTCB Disp SG Name
001f# 0006 0003 0006 0001 rdy 062f 7b7e4000 7b9c60d0 7b9ab1f8 1eb8 11 PMSHL32
# dp %1a060000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a060000* 0093e frame=00ele 0 0 c A U W P pageable
%1a060000 00ele frame=00ele 0 0 c A U W P pageable
#
```

>>> As expected the same page frame (00ele) is being referenced.

>>> Note also: frame ele of slot 8 is dirty (Dc=D) and unaccessed (Au=u) while in slot 1f it is clean and accessed. This tends to suggest that frame ele and therefore page %1a060000 was most recently updated by slot 8 and read by slot 1f before the update took place.

>>> Shared Arena, Instance Data:  
>>> -----

```
#.moc 5ef
*har par cpg va flg next prev link hash hob hal
01c6 %feb2170e 00000010 %1a890000 139 01c5 01c7 0000 0000 05ef 0000 =0000
hob har hobnxt flgs own hnte sown,cnt lt st xf
05ef 01c6 04e8 0024 050c 0131 0000 00 00 00 00 priv 0019 e:ipfc.exe
04e8 01c6 02ec 0024 04c6 0131 0000 00 00 00 00 priv 0018 c:epm.exe
02ec 01c6 0457 0024 049c 0131 0000 00 00 00 00 priv 0013 d:ibmavsd.exe
0457 01c6 0432 0024 0410 0131 0000 00 00 00 00 priv 0012 c:pmdraw.exe
0432 01c6 03d6 0024 0420 0131 0000 00 00 00 00 priv 0010 c:cmd.exe
03d6 01c6 0390 0024 0380 0131 0000 00 00 00 00 priv 000d c:pulse.exe
0390 01c6 03c8 0024 036c 0131 0000 00 00 00 00 priv 000c c:dinfo.exe
03c8 01c6 03a7 0024 034e 0131 0000 00 00 00 00 priv 000b c:mrfile32.exe
03a7 01c6 02c7 0024 0317 0131 0000 00 00 00 00 priv 000a c:pm diary.exe
02c7 01c6 0112 0024 02a6 0131 0000 00 00 00 00 priv 0006 c:pmshell.exe
0112 01c6 0000 0024 0091 0131 0000 00 00 00 00 priv 0003 c:pmshell.exe
```

>>> Object 5ef is an example of shared arena instance data. Each instance of the object has its own object record (VMOB), but they all share the same arena record. Each of these VMOBs is chained from the 'hob' field of the arena record via their 'hobnxt' field.  
>>> The VMOBs appear as private arena objects, but the arena record does not point to a specific hptda, which distinguishes this case as shared instance data.

>>> As would be expected with shared instance data the owners would differ for each instance object. They are infact the hptda's for each owner.

```
>>> The hmte's we would expect to be common to all the VMOBs.

#.mo 131
hob      va      flgs own  hmte  sown,cnt lt st xf
0131    %fe860978 8000 ffa6 0127 0000 00 00 00 00 mte      c:display.dll

>>> So, %1a890000 has been allocated by display.dll

>>> Again we can illustrate that we are really looking at instance data by
>>> examining the page tables of two examples:

.s8
Current slot number: 0008
# .p8
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0008# 0003 0001 0003 0001 rdy 061f 7b7b6000 7b9c484c 7b9a8a70 1eb8 01 PMSHL32
# dp %1a890000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a890000* 01291 frame=01066 0 0 D u s W P pageable
%1a890000 01066 frame=01066 0 0 D u s W P pageable
# .slf
Current slot number: 001f
# .plf
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
001f# 0006 0003 0006 0001 rdy 062f 7b7e4000 7b9c60d0 7b9ab1f8 1eb8 11 PMSHL32
# dp %1a890000
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a890000* 0093c frame=008e4 0 0 D u s W P pageable
%1a890000 008e4 frame=008e4 0 0 D u s W P pageable
#
>>> In PID 3, %1a890000 translates to physical address %01066000, but
>>> in PID 6, %1a890000 translates to physical address %008e4000.
```

## Finding Who Owns Memory

Having examined various types of arena, object and context record we now turn our attention to a more commonly asked question: "Who owns a particular location of memory?" To answer this we need to explore the match parameter of the .MA command.

.MAM search for arena records that encompass a given address:

```
# .mam %123456
har      par      cpg      va      flg next prev link hash hob      hal
008c %feblfc12 00000080 %00110000 169 00f1 0073 0000 0000 008f 0000 hptda=0091
023b %feb2211c 00001000 %000c0000 1e9 0238 0266 0000 0000 029f 0000 hptda=02a6
02fc %feb231b2 00000010 %00120000 169 02fd 02a2 0000 0000 0318 0000 hptda=036c
0306 %feb2328e 00000010 %00120000 169 0309 0321 0000 0000 03ba 0000 hptda=0380
0312 %feb23396 00000010 %00120000 169 0313 0311 0000 0000 03cd 0000 hptda=034e
032f %feb23614 00000080 %00110000 169 034b 02fa 0000 0000 03e4 0000 hptda=0317
0393 %feb23eac 00000010 %00120000 169 0394 038d 0000 0000 0452 0000 hptda=0410
0412 %feb24996 00000010 %00120000 169 0414 0411 0000 0000 04ef 0000 hptda=04c6
0517 %feb26004 00000010 %00120000 169 0519 0516 0000 0000 05f7 0000 hptda=050c
```

```
>>> Dump Formatter "hard-wires" the 'A' parameter so .mam=.mama (or .maam)
>>> Kernel Debugger needs 'A' explicitly if all contexts are to be searched.
>>> This only affects results from searching private arena addresses.
>>> Note: after fix pack 29 for Warp 3.0 and GA 4.0, .mam under the
>>> Dump Formatter behaves correctly. That is, the 'A' parameter is no longer
>>> "hard-wired".
```

```
>>> We can also add the 'C' parameter to chain through the related VMOBs
>>> and VMCOs at the same time.
```

```
# .mamc %123456

*har      par      cpg      va      flg next prev link hash hob      hal
```

```

008c %feblfc12 00000080 %00110000 169 00f1 0073 0000 0000 008f 0000 hptda=0091
hob har hobnxt flgs own hmte sown,cnt lt st xf
008f 008c 0000 422c 0091 01c0 0000 00 00 00 00 priv 0003 c:pmsshell.exe

*har par cpg va flg next prev link hash hob hal
023b %feb2211c 00001000 %000c0000 1e9 0238 0266 0000 0000 029f 0000 hptda=02a6
hob har hobnxt flgs own hmte sown,cnt lt st xf
029f 023b 0000 423c 02a6 01d7 0000 00 00 00 00 priv 0006 c:pmsshell.exe

*har par cpg va flg next prev link hash hob hal
02fc %feb231b2 00000010 %00120000 169 02fd 02a2 0000 0000 0318 0000 hptda=036c
hob har hobnxt flgs own hmte sown,cnt lt st xf
0318 02fc 0000 422c 036c 0371 0000 00 00 00 00 priv 000c c:dinfo.exe

*har par cpg va flg next prev link hash hob hal
0306 %feb2328e 00000010 %00120000 169 0309 0321 0000 0000 03ba 0000 hptda=0380
hob har hobnxt flgs own hmte sown,cnt lt st xf
03ba 0306 0000 422c 0380 035c 0000 00 00 00 00 priv 000d c:pulse.exe

*har par cpg va flg next prev link hash hob hal
0312 %feb23396 00000010 %00120000 169 0313 0311 0000 0000 03cd 0000 hptda=034e
hob har hobnxt flgs own hmte sown,cnt lt st xf
03cd 0312 0000 422c 034e 0354 0000 00 00 00 00 priv 000b c:mrfile32.exe

*har par cpg va flg next prev link hash hob hal
032f %feb23614 00000080 %00110000 169 034b 02fa 0000 0000 03e4 0000 hptda=0317
hob har hobnxt flgs own hmte sown,cnt lt st xf
03e4 032f 0000 422c 0317 01c0 0000 00 00 00 00 priv 000a c:pmdiary.exe

*har par cpg va flg next prev link hash hob hal
0393 %feb23eac 00000010 %00120000 169 0394 038d 0000 0000 0452 0000 hptda=0410
hob har hobnxt flgs own hmte sown,cnt lt st xf
0452 0393 0000 402c 0410 ff3e 0000 00 00 00 00 priv 0012 c:pmdraw.exe

*har par cpg va flg next prev link hash hob hal
0412 %feb24996 00000010 %00120000 169 0414 0411 0000 0000 04ef 0000 hptda=04c6
hob har hobnxt flgs own hmte sown,cnt lt st xf
04ef 0412 0000 422c 04c6 04f3 0000 00 00 00 00 priv 0018 c:epm.exe

*har par cpg va flg next prev link hash hob hal
0517 %feb26004 00000010 %00120000 169 0519 0516 0000 0000 05f7 0000 hptda=050c
hob har hobnxt flgs own hmte sown,cnt lt st xf
05f7 0517 0000 422c 050c 05de 0000 00 00 00 00 priv 0019 e:ipfc.exe

```

```

>>> .MAMC is such a frequently used command that it is made the default
>>> specification for .M
>>> Further more, .M will take the default CS:EIP as the match
>>> address if no address is given.

```

```

>>> Suppose we wish to find out what code is being currently executed in
>>> in slot 39...

```

```

# .s 39
Current slot number: 0039
# .p #
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0039 0019 0010 0019 0001 rdy 061f 7b818000 7b9ca230 7b9adea8 1f0c 12 IPFC
# .r
eax=00000000 ebx=00307d90 ecx=00320000 edx=00000000 esi=00001000 edi=00001000
eip=1a022240 esp=0004d098 ebp=0004d0b4 iopl=2 -- -- -- nv up ei pl nz na pe nc
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001d6000
005b:1a022240 83c418 add esp,+18
# ln
No Symbols Found
# .m

*har par cpg va flg next prev link hash hob hal
00dd %feb20308 00000010 %1a020000 3d9 00dc 00de 0000 0000 00ea 0000 hco=007ba
hob har hobnxt flgs own hmte sown,cnt lt st xf
00ea 00dd 0000 0838 00e5 00e5 0000 00 00 00 00 shared c:doscall11.dll
hco=07ba pco=fe6806bd hconext=00822 hptda=050c f=1c pid=0019 e:ipfc.exe

```

```

>>> The current cs:eip for slot 39 is executing in doscall11.dll and
>>> has been called either directly or indirectly by ipfc.exe

```

Finally in this section we answer, "What is the hptda given the PTDA address?"

This required the use of the match parameter with .MO.

.MOM is more restrictive and .MAM. It will only return a result if the supplied address is a precise match for the beginning of a pseudo-object. Since the PTDA is a pseudo-object we can use its address with .MOM:

```
# .p 2a
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
002a  0006  0003  0006  000c  blk  021f  7b7fa000  7b9c60d0  7b9ac4e0  leac  11  PMSHL32
# .mom %7b9c60d0
hob      va      flgs  own  hmte  sown,cnt  lt  st  xf
02a6    %7b9c60d0  8000  ffc b 0000  0000 00  00 00 00 ptda 0006 c:pmshell.exe

>>> The hptda for Pid 6 is therefore 2a6.
```

## How to Correlate Named Memory With its Address

Here's how to locate named shared memory, answer who's sharing it and whether a particular address in the shared arena is named.

Named memory is managed using a [RMP](#) (Record Management Package). This is a generalised kernel facility for managing global data of variable length. The **RMP** facility provides allocation, deletion, add and find services. Each **RMP** user references his **RMP** structure using a handle. The **RMP** itself is limited to a maximum of 64K.

>>> First locate the handle of named shared memory's RMP:

```
##dw sharermpstruc l2
0400:00004506 0004 0178
|||
|| ..... Selector for RMP segment
..... Flags xxxxxxxx
.....1 = Segment busy
.....1. = Somebody's waiting
.....1.. = Segment allocated
```

>>> The RMP handle is used as the blockid (by ProcBlock) for serialising  
>>> RMP manipulations.

>>>

>>> Now display the named memory RMP segment:

```
##db 178:0
0178:00000000 00 06 a2 03 5e 02 5e 02-03 00 00 00 00 00 00 00 ..".^..^.....
0178:00000010 83 ff 00 00 11 00 00 00-9b 06 01 00 44 4f 53 5c .....DOS\
0178:00000020 43 44 49 42 00 13 00 88-01 47 bd 04 00 50 4d 44 CDIB....G=..PMD
0178:00000030 52 41 47 2e 4d 45 4d 00-08 00 9f 00 88 01 01 00 RAG.MEM.....
0178:00000040 14 00 91 01 d7 bc 02 00-53 4d 47 5c 53 47 54 49 ....W<..SMG\SGTI
0178:00000050 54 4c 45 00 08 00 9f 00-91 01 01 00 12 00 93 01 TLE.....
0178:00000060 b7 bc 02 00 42 56 53 5c-42 56 53 30 30 00 08 00 7<..BVS\BVS00...
0178:00000070 9f 00 93 01 01 00 12 00-a8 01 8f bc 01 00 42 56 .....(<...BV
##d
0178:00000080 53 5c 42 56 53 30 31 00-08 00 9f 00 a8 01 01 00 S\BVS01.....(...
0178:00000090 12 00 ab 01 67 bc 01 00-42 56 53 5c 42 56 53 30 ..+.g<..BVS\BVS0
0178:000000a0 33 00 08 00 9f 00 ab 01-01 00 18 00 c4 01 3f bc 3.....+.....D.?<
0178:000000b0 02 00 53 4d 47 5c 50 4d-48 44 45 52 52 2e 44 41 ..SMG\PMHDERR.DA
0178:000000c0 54 00 08 00 af 01 c4 01-01 00 08 00 af 01 91 01 T.../.D...../...
0178:000000d0 01 00 08 00 af 01 93 01-01 00 08 00 9f 00 c4 01 .../.....D.
0178:000000e0 01 00 12 00 43 02 2f a0-02 00 42 56 53 5c 42 56 ....C./ ..BVS\BV
0178:000000f0 53 31 30 00 08 00 9f 00-43 02 01 00 08 00 4c 02 S10.....C.....L.
##d
0178:00000100 43 02 01 00 08 00 58 02-88 01 01 00 17 00 9c 02 C.....X.....
0178:00000110 cf 9f 01 00 50 4d 57 50-5c 43 4c 41 53 53 2e 54 O...PMWP\CLASS.T
0178:00000120 42 4c 00 08 00 58 02 9c-02 01 00 08 00 fa 02 88 BL...X.....z..
0178:00000130 01 01 00 18 00 13 04 e7-9e 01 00 45 50 4d 5c 45 .....g...EPM\E
0178:00000140 54 4b 45 36 30 30 2e 45-50 4d 00 08 00 fa 02 13 TKE600.EPM...z...
0178:00000150 04 01 00 10 00 15 04 ff-9d 01 00 45 50 4d 47 4e .....EPMGN
0178:00000160 4c 53 00 08 00 fa 02 15-04 01 00 18 00 03 04 e7 LS...z.....g
0178:00000170 9d 01 00 31 35 35 30 32-33 33 33 5c 45 50 4d 2e ...15502333\EPM.
```

```

##d
0178:00000180 45 58 00 08 00 fa 02 03-04 01 00 08 00 56 03 88 EX...z.....V..
0178:00000190 01 01 00 1b 00 f6 02 bf-9d 01 00 31 35 33 39 34 ....v.?...15394
0178:000001a0 39 31 34 5c 45 33 45 4d-55 4c 2e 45 58 00 08 00 914\E3EMUL.EX...
0178:000001b0 fa 02 f6 02 01 00 12 00-90 03 8f 9d 03 00 42 56 z.v.....BV
0178:000001c0 53 5c 42 56 53 31 34 00-08 00 9f 00 90 03 01 00 S\BVS14.....
0178:000001d0 08 00 32 04 90 03 01 00-12 00 43 04 6f 9d 03 00 ..2.....C.o...
0178:000001e0 42 56 53 5c 42 56 53 31-35 00 08 00 9f 00 43 04 BVS\BVS15.....C.
0178:000001f0 01 00 08 00 48 04 43 04-01 00 08 00 57 04 90 03 ....H.C.....W...
##d
0178:00000200 01 00 08 00 62 04 43 04-01 00 12 00 87 04 4f 9d ....b.C.....O.
0178:00000210 03 00 42 56 53 5c 42 56-53 31 36 00 08 00 9f 00 ..BVS\BVS16.....
0178:00000220 87 04 01 00 08 00 8b 04-87 04 01 00 08 00 99 04 .....
0178:00000230 87 04 01 00 12 00 0a 03-ef 9c 03 00 42 56 53 5c .....o...BVS\
0178:00000240 42 56 53 31 38 00 08 00-9f 00 0a 03 01 00 08 00 BVS18.....
0178:00000250 d6 04 0a 03 01 00 08 00-bb 04 0a 03 01 00 a2 83 V.....;.....".
0178:00000260 00 00 5e 02 00 00 9a 83-00 00 66 02 00 00 00 00 ..^.....f.....
0178:00000270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....

>>> The first 20 bytes form the RMP header, the remainder is a series of
>>> variable length records.
>>> Examining the header first we have:
>>> +00 00 06      = total size of segment (0600)
>>> +02 a2 03      = amount of free space (03a2)
>>> +04 5e 02      = link to first free block (025e)
>>> +06 5e 02      = start of last free block (025e)
>>> +08 03 00 00 00 = heap handle (0003 is kernel heap handle from which
>>>                  RMP is alloc'd)
>>> +0c 00 00 00 00 = PG alloc/realloc flags
>>> +10 83 ff      = hobowner (handle of user of this RMP is ff83)
>>> +12 00 00      = hobmte (hmte of user of this RMP. It's the kernel so 0000)

>>> Check out the owner of this RMP

##.mo ff83
ff83 mshrmp

>>> 'mshrmp' is named shared memory management

>>> Records follow the header. They are prefixed by a word length that includes
>>> 2 bytes for the length field itself. If the record is free then the high
>>> order bit of the length is set. The data within the record is private to
>>> the owner.

>>> The first record in this RMP is:

          ..... length 0011
          || ||
>>> 0178:00000010 .. .. .. 11 00 00 00-9b 06 01 00 44 4f 53 5c
>>> 0178:00000020 43 44 49 42 00 .. .. ..

>>> The second record in this RMP is:

>>> 0178:00000020 .. .. .. 13 00 88-01 47 bd 04 00 50 4d 44
>>> 0178:00000030 52 41 47 2e 4d 45 4d 00-.. .. .. ..

>>> Named shared memory management uses two forms of RMP record:
>>>   Global - to keep the name, handle, selector and total reference count
>>>   Local  - One for each process sharing the named memory. Contains
>>>             hptda, hob and ref count for within the given process.
>>>
>>> Breaking down record 2 we have:

>>> +00 0013      length of record
>>> +02 0188      hob of shared object
>>> +04 bd47      selector of shared object
>>> +06 0004      reference count
>>> +08 PMDRAG.MEM name (with \SHAREMEM\ prefix omitted) and terminated with
>>>                  a null byte (that is, it's an ASCII string)

>>> Check out hob 188
##.moc 188

*har      par      cpgr      va      flg next prev link hash hob      hal
0140 %fecffb8a 00000010 %17a80000 369 000f 0141 0000 0000 0188 0000 hco=00291
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0188 0140 0000 482c ff82 017d 0000 00 00 00 00 mshare
hco=00291 pco=fe85dcf0 hconext=003c8 hptda=02fa f=16 pid=0013 c:epm.exe
hco=003c8 pco=fe85e303 hconext=00246 hptda=0356 f=16 pid=0009 c:mrfile32.exe

```



```

hco=00246 pco=fe85db79 hconext=00070 hptda=0258 f=16 pid=0005 c:pmsshell.exe
hco=00070 pco=fe85d24b hconext=00000 hptda=009f f=17 pid=0002 c:pmsshell.exe

```

```

>>> We see 4 owners in accordance with the reference count
>>> note: the owner of the object is 'mshare'

```

```

>>> Check out the selector in record 2:

```

```

##d1 bd47
bd47 Data Bas=17a80000 Lim=00000067 DPL=3 P RW A

```

```

>>> In this case it's within the compatibility region so could have used the
>>> CRMA to get %17a80000 directly

```

```

>>> The processes sharing the named storage may be obtained directly from local
>>> records in the RMP. A local record is of the following form:

```

```

>>>
>>> +00 word - length of record (always 0008)
>>> +02 word - hptda of user
>>> +04 word - handle of shared memory object
>>> +06 word - reference count for this ptda.

```

```

>>> Scanning through the RMP (for object 0188) we find the following local
>>> records:

```

```

>>> 0178:00000030 .. .. .. .. -08 00 9f 00 88 01 01 00
>>> 0178:00000100 .. .. .. .. 08 00 58 02-88 01 01 00 .. .. ..
>>> 0178:00000120 .. .. .. .. -.. .. .. 08 00 fa 02 88
>>> 0178:00000130 01 01 00 .. .. .. .. -.. .. .. ..
>>> 0178:00000180 .. .. .. .. -.. .. .. 08 00 56 03 88
>>> 0178:00000190 01 01 00 .. .. .. .. -.. .. .. ..

```

```

>>> This confirms what was shown in the .mo 188 display, but we have in addition
>>> the reference count for each process.

```

```

>>> Finally, we can cut the cake a different way by asking what is all the
>>> named storage being referenced by a particular process. For example
>>> EPM.
>>> Start by finding its slot nos.

```

```

.p
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
.
.
.
0027 0013 0002 0013 0001 blk 0200 7b974000 7bb460d0 7bb2bf24 1eb8 12 epm
0020 0013 0002 0013 0002 blk 0200 7b966000 7bb460d0 7bb2b338 1ecc 12 epm
.
.

```

```

>>> EPM's pPTDA is %7bb460d0. Now find the hptda of this PTDA

```

```

##.mom %7bb460d0
hob va flgs own hmtc sown,cnt lt st xf
02fa %7bb460d0 8000 ffc3 035b 0000 00 00 00 00 ptda 0013 c:epm.exe

```

```

>>> Answer: 2fa.
>>> Now look through the RMP for local records that begin:
>>> 08 00 fa 02

```

```

>>> 0178:00000120 .. .. .. .. -.. .. .. 08 00 fa 02 88
>>> 0178:00000130 01 01 00 .. .. .. .. -.. .. .. ..
>>> 0178:00000140 .. .. .. .. -.. .. .. 08 00 fa 02 13
>>> 0178:00000150 04 01 00 .. .. .. .. -.. .. .. ..
>>> 0178:00000160 .. .. .. 08 00 fa 02 15-04 01 00 .. .. ..
>>> 0178:00000180 .. .. .. 08 00 fa 02 03-04 01 00 .. .. ..
>>> 0178:000001a0 .. .. .. .. -.. .. .. .. 08 00
>>> 0178:000001b0 fa 02 f6 02 01 00 .. -.. .. .. ..

```

```

>>> So, 5 named objects, with hobs=0188, 0413, 0415, 0403 and 02f6

```

```
>>> Scanning the the RMP for Global records for these objects will reveal
>>> their names: PMDRAG.MEM, EPM\ETKE600.EPM, EPMGNLS, 15502333\EPM.EXE,
>>> 15394914\E3EMUL.EX.

>>> issuing .mo against each of the hobs will reveal whether these are shared
>>> and with whom:

.moc 188

*har      par      cpg      va      flg next prev link hash hob      hal
0140 %fecffb8a 00000010 %17a80000 369 000f 0141 0000 0000 0188 0000 hco=00291
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0188 0140 0000 482c ff82 017d 0000 00 00 00 00 mshare
hco=00291 pco=fe85dcf0 hconext=003c8 hptda=02fa f=16 pid=0013 c:epm.exe
hco=003c8 pco=fe85e303 hconext=00246 hptda=0356 f=16 pid=0009 c:mrfile32.exe
hco=00246 pco=fe85db79 hconext=00070 hptda=0258 f=16 pid=0005 c:pmsshell.exe
hco=00070 pco=fe85d24b hconext=00000 hptda=009f f=17 pid=0002 c:pmsshell.exe
##.moc 413

*har      par      cpg      va      flg next prev link hash hob      hal
0367 %fed02ae4 00000010 %13dc0000 369 025e 0372 0000 0000 0413 0000 hco=00293
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0413 0367 0000 4a2c ff82 030c 0000 00 00 00 00 mshare
hco=00293 pco=fe85dcfa hconext=00000 hptda=02fa f=17 pid=0013 c:epm.exe
##.moc 415

*har      par      cpg      va      flg next prev link hash hob      hal
0307 %fed022a4 00000010 %13bf0000 369 037e 02c2 0000 0000 0415 0000 hco=003cc
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0415 0307 0000 4a2c ff82 0407 0000 00 00 00 00 mshare
hco=003cc pco=fe85e317 hconext=00000 hptda=02fa f=17 pid=0013 c:epm.exe
##.moc 403

*har      par      cpg      va      flg next prev link hash hob      hal
02c2 %fed01cb6 00000030 %13bc0000 369 0307 0262 0000 0000 0403 0000 hco=00309
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0403 02c2 0000 4a2c ff82 0407 0000 00 00 00 00 mshare
hco=00309 pco=fe85df48 hconext=00000 hptda=02fa f=17 pid=0013 c:epm.exe
##.moc 2f6

*har      par      cpg      va      flg next prev link hash hob      hal
0273 %fed015ec 00000010 %13b70000 369 027c 02ed 0000 0000 02f6 0000 hco=00308
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
02f6 0273 0000 4a2c ff82 0407 0000 00 00 00 00 mshare
hco=00308 pco=fe85df43 hconext=00000 hptda=02fa f=17 pid=0013 c:epm.exe
##

>>> We see that except for hob=188 all the others are for the private use of
>>> EPM.
```

## How Memory Aliasing Works

Aliasing is a facility in virtual memory management whereby one or more pages of a memory object may be referenced from an alternative virtual address, possibly from a different process or arena and possibly with different read/write/execute characteristics. It is used extensively by device drivers debugging applications and VDMs.

This example shows how aliasing is represented in the system for a debugging application and how shared storage becomes privatized. There are many ways of creating aliases. The application in this example is IPMD, which uses DosDebug function MapWRAlias to alias the debuggee's storage and DosCreateCSAlias to map a code selector to one of his own data segments.

We introduce the [memory alias record \(VMAL\)](#) and the [.ML command](#).

```
>>> For reference list the thread slots in the system...

.p
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
```

```
.
001c 0004 0002 0004 0001 blk 0200 7b95e000 7bb45078 7bb2ac68      10 cmd
.
.
0020 0008 0002 0008 0002 blk 0200 7b966000 7bb460d0 7bb2b338      12 mrfile32
002b# 000b 0002 000b 0001 blk 0200 7b97c000 7bb468fc 7bb2c5f4 1eb8 14 ipmd
002a 000b 0002 000b 0002 blk 0200 7b97a000 7bb468fc 7bb2c440 1eb8 14 ipmd
002c 000d 0002 000d 0001 blk 0200 7b97e000 7bb47954 7bb2c7a8 1eb8 16 cmd
002d 000c 0002 000c 0001 blk 0200 7b980000 7bb47128 7bb2c95c 1e98 15 dpmlines
002e 000e 0002 000e 0001 blk 0300 7b982000 7bb48180 7bb2cb10 1eb8 17 epm
002f 000e 0002 000e 0002 blk 0300 7b984000 7bb48180 7bb2ccc4 1ecc 17 epm
```

>>> Now list all the busy alias records:

```
##.ml
hal=0001 pal=%ffe61020 har=00b8 hptda=009f pgoff=00000 f=001
hal=0002 pal=%ffe61028 har=00b9 hptda=009f pgoff=00000 f=001
hal=0003 pal=%ffe61030 har=001b hptda=009f pgoff=00000 f=001
hal=0004 pal=%ffe61038 har=0183 cs=00e6 ds=d446 cref=001 f=13
hal=0005 pal=%ffe61040 har=0199 hptda=009f pgoff=00006 f=001
hal=0006 pal=%ffe61048 har=01b8 hptda=009f pgoff=00000 f=021
hal=0007 pal=%ffe61050 har=01b9 hptda=009f pgoff=00000 f=021
hal=0008 pal=%ffe61058 har=01ba hptda=009f pgoff=00000 f=021
hal=0009 pal=%ffe61060 har=01e7 hptda=009f pgoff=00000 f=001
hal=000a pal=%ffe61068 har=0208 cs=0056 ds=d446 cref=001 f=13
hal=000b pal=%ffe61070 har=020b cs=0056 ds=d446 cref=001 f=13
hal=000c pal=%ffe61078 har=026f cs=007e ds=d446 cref=001 f=13
hal=000d pal=%ffe61080 har=02bf cs=00ae ds=d446 cref=001 f=13
hal=000e pal=%ffe61088 har=02df cs=01ae ds=0077 cref=001 f=13
hal=000f pal=%ffe61090 har=0305 hptda=0389 pgoff=00000 f=049
hal=0010 pal=%ffe61098 har=0306 hptda=0389 pgoff=00000 f=049
hal=0011 pal=%ffe610a0 har=030e cs=0056 ds=d446 cref=001 f=13
hal=0012 pal=%ffe610a8 har=0323 cs=0056 ds=d446 cref=001 f=13
hal=0013 pal=%ffe610b0 har=032e cs=007e ds=d446 cref=001 f=13
```

```
>>> hal f & 10 have f=049 = 0000 0100 1001
>>>
>>>
>>>
>>>
>>>
>>> har=305 is an alias for a linear address in hptda=389
>>> similarly har=306 is an alias for a linear address in hptda=389
>>> Now look closer at hal=f.
```

```
##.mlc f
>>> chaining doesn't always work so..
```

```
##.mac 305
```

```
*har      par      cpgr      va      flg next prev link hash hob      hal
0305 %fed02278 00000010 %00520000 169 0307 0304 00b5 0000 00c1 000f hptda=031a
hal=000f pal=%ffe61090 har=0305 hptda=0389 pgoff=00000 f=049
har      par      cpgr      va      flg next prev link hash hob      hal
00b5 %fecfef98 00000010 %1a030000 3d9 00b4 00b6 0000 0000 00c1 0000 hco=00502
hob      har hobnxt flgs own hmtc sown,cnt lt st xf
00c1 0305 0000 1838 00bd 00bd 0000 00 00 00 00 shared c:doscall1.dll
hco=00502 pco=fe85e925 hconext=00473 hptda=03cb f=1c pid=000e c:epm.exe
hco=00473 pco=fe85e65a hconext=00455 hptda=03b9 f=1c pid=000d c:cmd.exe
hco=00455 pco=fe85e5c4 hconext=0045a hptda=0389 f=9c pid=000c d:dpmlines.exe
hco=0045a pco=fe85e5dd hconext=00283 hptda=031a f=1c pid=000b d:ipmd.exe
hco=00283 pco=fe85dcaa hconext=0014a hptda=02d6 f=1c pid=0008 c:mrfile32.exe
hco=0014a pco=fe85d68d hconext=00133 hptda=0257 f=1c pid=0005 c:pmshell.exe
hco=00133 pco=fe85d61a hconext=00092 hptda=0248 f=1c pid=0004 c:cmd.exe
hco=00092 pco=fe85d2f5 hconext=00020 hptda=01ae f=1c pid=0003 c:harderr.exe
hco=00020 pco=fe85d0bb hconext=00000 hptda=009f f=1c pid=0002 c:pmshell.exe
```

```
>>> Ignoring hco=455 for the moment. har=b5 represents linear address range
>>> %1a030000 in the shared arena. This is in fact a shared object (hob=1c)
>>> which is being accessed by 9 different processes. The hco chain lists those
>>> processes that access this object. hob=1c is one of the objects in doscall1.dll
```

```
>>> Let's verify that %1a030000 is indeed that same data in each of the contexts.
>>> hco=133 is for pid=4, slot=1c, cmd.exe(1)
##.s 1c
##dp %1a030000 12
```

linaddr	frame	pteframe	state	res	Dc	Au	CD	WT	Us	rW	Pn	state
%1a030000*	00236	frame=00236	2	0	D	A			U	W	P	resident
%1a030000	00a22	frame=00a22	0	0	c	u			U	r	P	pageable
%1a031000	00alc	vp id=00320	0	0	c	u			U	r	n	pageable

>>> %1a030000 equates to real address %a22000  
>>> hco=473 is for pid=d, slot=2c, cmd.exe (2)

```
##.s 2c
##dp %1a030000 12
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a030000* 00c3c frame=00c3c 2 0 D A U W P resident
%1a030000 00a22 frame=00a22 0 0 c u U r P pageable
%1a031000 00alc frame=00alc 0 0 c u U r P pageable
```

>>> %1a030000 equates to real address %a22000  
>>> hco=502 is for pid=e, slot=2e, epm.exe

```
##.s 2e
##dp %1a030000 12
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a030000* 00418 frame=00418 2 0 D A U W P resident
%1a030000 00a22 frame=00a22 0 0 c u U r P pageable
%1a031000 00alc frame=00alc 0 0 c u U r P pageable
```

>>> In each of these cases looked so far, linear address %1a030000 is  
>>> mapped to the same real address %a22000.

>>> Now examine the hco flags: 9c = 1001 1100

```
>>> | | |.... User
>>> | | |..... Executable
>>> | | |..... Read
>>> | | |..... Privatized
```

>>> now turn our attention to har=305, hal=f, address %520000 and hco=455.

>>> hco=455 has the additional Privatized flag set.  
>>> Now look at the PTE for this storage in slot=2d, pid=c, dpmlines.exe

```
##.s 2d
##dp %1a030000 12
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a030000* 005ab frame=005ab 2 0 D A U W P resident
%1a030000 00bda frame=00bda 0 0 c u U r P pageable
%1a031000 vp id=01616 0 0 c u U r n pageable
```

>>> This is no-longer the same storage as in the other contexts. After DPMLINES  
>>> was loaded, IPMD created an alias to object 1c reference by DMPLINES.  
>>> The loader/memory had to make a private copy to protect the integrity  
>>> of other contexts who were sharing the same object. Having privatized  
>>> this object for the one context, the loader will not share it with  
>>> other contexts.  
>>> If we hadn't started with the alias record we could have done a .ml now  
>>> and looked for the records which referenced hptda=389. As it happens we  
>>> know already that har=305 is an alias of har=b5. We can check this out  
>>> by looking at the page tables for %520000 in hptda=031a (pid=b, slot=2c)

```
##.s 2c
##dp %520000
##.i %520000
##dp %520000 12
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00520000* 00390 frame=00390 2 0 D A U W P resident
%00520000 00bda frame=00bda 0 0 c u U W P pageable
%00521000 vp id=01616 0 0 c u U W n pageable
```

>>> %520000 is %bda000 which is the same real address as %1a030000 in slot=  
>>> 2d. We had to page in %520000 so we should check %1a030000 in slot=2d  
>>> again, in case it was discarded.

```
##.s 2d
##dp %1a030000 12
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%1a030000* 005ab frame=005ab 2 0 D A U W P resident
%1a030000 00bda frame=00bda 0 0 c u U r P pageable
%1a031000 vp id=01616 0 0 c u U r n pageable
```

> ... and it is the same.  
> While we are at it, lets check %1a030000 in slot=2c (IPMD)

```
##.s 2c
##dp %1a030000 12
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
```

```
%1a030000* 00611 frame=00611 2 0 D A U W P resident
%1a030000 00a22 frame=00a22 0 0 c u U r P pageable
%1a031000 00alc vp id=00320 0 0 c u U r n pageable
```

```
>>> ... and yes as expected IPMD is referencing the shared %1a030000, in
>>> fact he is referencing both copies.
```

```
>>> Now let's look at some of the other aliases set up by IPMD
##.mlc 10
```

```
*har par cpg va flg next prev link hash hob hal
0306 %fed0228e 00000010 %00540000 169 0308 0307 02ee 0000 0386 0010 hptda=031a
hal=0010 pal=%ffe61098 har=0306 hptda=0389 pgoff=00000 f=049
har par cpg va flg next prev link hash hob hal
02ee %fed0207e 00000010 %00010000 1d9 02ed 02f0 0000 0000 0386 ffff hptda=0389
```

```
>>> Note: hal=ffff for har=2ee. This is a special hal to indicate a
>>> privatized arena - there isn't a context record to put the privatized
>>> flag in as this was private arena, shared data.
```

```
>>> Check out the hptda pids as we have forgotten who 31a and 389 are..
```

```
##.mo 31a
hob va flgs own hmte sown,cnt lt st xf
031a %7bb468fc 8000 ffc b 02db 0000 00 00 00 00 ptda 000b d:ipmd.exe
##.mo 389
hob va flgs own hmte sown,cnt lt st xf
0389 %7bb47128 8000 ffc b 0000 0000 00 00 00 00 ptda 000c d:dpmlines.exe
```

```
>>> Now check the page tables in each processes to prove we are looking
>>> at the same thing...
```

```
##.ss 2b
##dp %540000 11
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00540000* 00390 frame=00390 2 0 D A U W P resident
%00540000 vp id=015f0 0 0 c u U W n pageable
##.i %540000
##dp %540000 11
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00540000* 00390 frame=00390 2 0 D A U W P resident
%00540000 005d2 frame=005d2 0 0 c u U W P pageable
##.ss 2d
##dp %10000 12
##.i %10000
##dp %10000 12
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00010000* 002b3 frame=002b3 2 0 D A U W P resident
%00010000 005d2 frame=005d2 0 0 c u U r P pageable
%00011000 vp id=015f1 0 0 c u U r n pageable
##.ss 2b
##dp %540000 11
linaddr frame pteframe state res Dc Au CD WT Us rW Pn state
%00540000* 00390 frame=00390 2 0 D A U W P resident
%00540000 005d2 frame=005d2 0 0 c u U W P pageable
```

```
>>> Finally look alias record hal=e. This is a CS Alias of a data
>>> segment within in the same process. The hal flags indicate:
```

```
>>> 13=0001 0011
>>> | | |... Busy (in use)
>>> | | |.... CS Alias
>>> | ..... DS selector valid
```

```
##.mlc e
```

```
*har par cpg va flg next prev link hash hob hal
02df %fed01f34 00000010 %00350000 1c9 02e2 02de 029d 0000 031b 000e hptda=031a
hal=000e pal=%ffe61088 har=02df cs=01ae ds=0077 cref=001 f=13
har par cpg va flg next prev link hash hob hal
029d %fed01988 00000010 %000e0000 179 02b9 0293 0000 0000 031b 0000 hptda=031a
hob har hobnxt flgs own hmte sown,cnt lt st xf
031b 02df 0000 102c 031a 031e 0000 00 00 00 00 priv 000b d:ipmd.exe
```

```
>>> Check out the page tables...
```

```
##dp %350000 12
##.i %350000
##dp %350000 12
```

```

linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%00350000* 00625 frame=00625 2    0  D  A          U  W  P  resident
%00350000 00362 frame=00362 0    0  c  u          U  r  P  pageable
%00351000   vp id=01403 0    0  c  u          U  r  n  pageable
##dp %e0000 12
linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%000e0000* 00625 frame=00625 2    0  D  A          U  W  P  resident
%000e0000 00362 vp id=01402 0    0  c  u          U  W  n  pageable
%000e1000   vp id=01403 0    0  c  u          U  W  n  pageable
##.i %e0000
##dp %e0000 12
linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%000e0000* 00625 frame=00625 2    0  D  A          U  W  P  resident
%000e0000 00362 frame=00362 0    0  c  u          U  W  P  pageable
%000e1000   vp id=01403 0    0  c  u          U  W  n  pageable
##dp %350000 12
linaddr  frame  pteframe  state  res  Dc  Au  CD  WT  Us  rW  Pn  state
%00350000* 00625 frame=00625 2    0  D  A          U  W  P  resident
%00350000 00362 frame=00362 0    0  c  u          U  r  P  pageable
%00351000   vp id=01403 0    0  c  u          U  r  n  pageable

>>> Check out the segment descriptors ....
##dl lae
01ae Code   Bas=00350000 Lim=0000f15f DPL=2 P  RE C
##dl 77
0077 Data   Bas=000e0000 Lim=0000f15f DPL=3 P  RW   A
##

```

```

>>> Beacuse of the existence of the CS/DS alias, IPMD can effectively
>>> read, write and execute the same storage. This is how IPMD is able
>>> to implement break points, by copying code, patching in INT 3
>>> instructions and executing the copied code; all from ring 3 priviledge
>>> without compromising other processes or the system.

```

## Exploring 32-bit Presentation Manager Under WARP

In this section we look specifically at the messaging function within Presentation Manager (PM).

Sending and receiving messages lies at the heart of how PM applications communicate with each other and the system. Messages may be sent synchronously and posted asynchronously. The mismanagement of messages by an application leads frequently to the 'Bad Application' Pop-up dialog. In extreme cases deadlocks result.

This topic applies to OS/2 V3, which introduced the 32-bit version of Presentation Manager. The previous 16-bit environment has analogous concepts which are briefly explored through a final worked example.

Pre-requisites to any PM debugging requires the following:

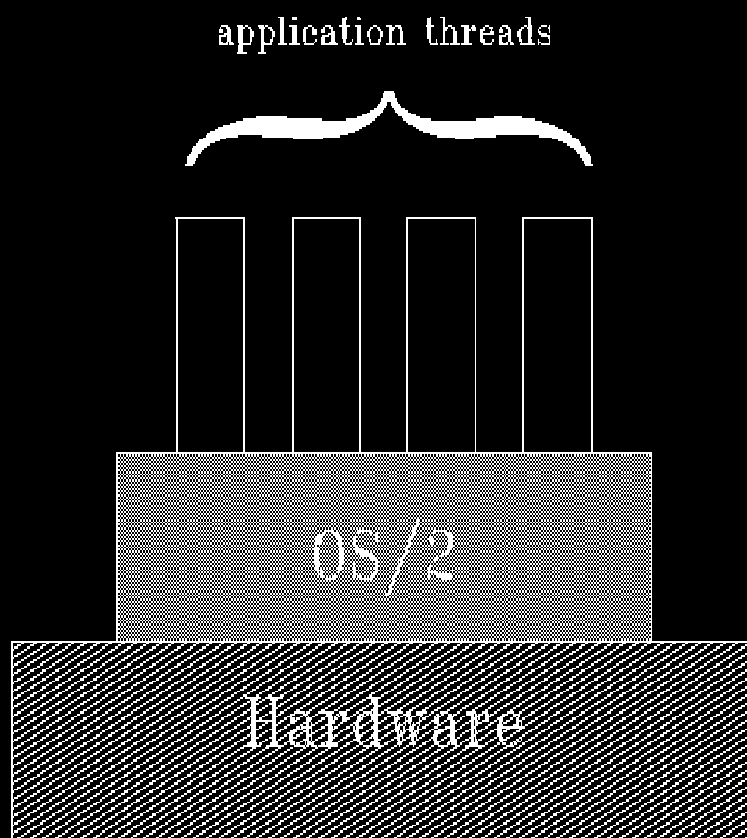
- Availability of the symbol file (**PMMERGE.SYM**) for **PMMERGE.DLL**. This should be installed in the same directory as **PMMERGE.DLL** when using the Kernel Debugger or for dump analysis, in the same directory as the Dump Formatter.
- Availability of the PM programming reference from the Programmer's ToolKit.
- Also of use, is ready access to the C header files from the Programmer's Toolkit.

We start by giving an overview of the PM messaging environment in which an application's PM thread operates.

## The PM Messaging Environment

First consider the non-PM application programming model as shown in the following diagram:

# Non-PM Application Program Model



This diagram illustrates the following points:

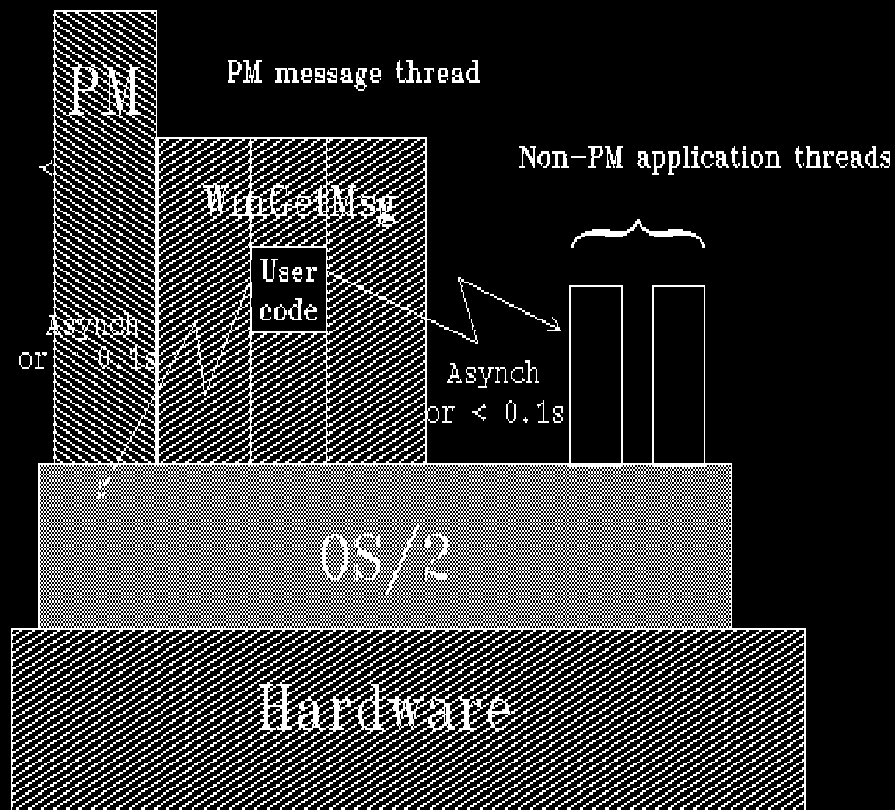
- Non-PM application threads run in a relatively unconstrained environment (compare this with the following situation).
- The Operating System provides a *black-box* set of services and interfaces.
- The Hardware is not directly accessible by the application.

For PM message threads, the environment is radically different. The key difference is that application code that runs on a PM message thread is effectively a subroutine of the **WinGetMsg** API even though **WinGetMsg** is called by the application. The terminology often used to describe this reversal is *Program Inversion*. **WinGetMsg** is said to be *inverted* with respect to the application's message thread.

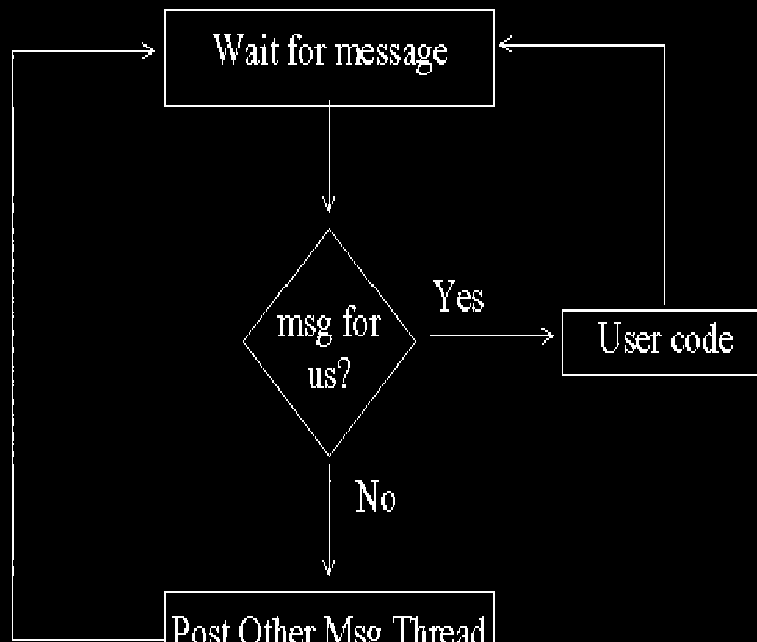
We see this illustrated in the following diagram.



# PM Application Program Model



## Msg Thread Logic



Also illustrated by this diagram are the following points:

- PM Message threads act in a co-operative way. They wait for messages, and *pass them on* to the appropriate application if not for themselves.
- PM Message threads should spend most of their elapsed time waiting for notification of messages - because of their co-operative nature.
- Application code that runs on the message thread should be limited to very short duration processing. We often speak of the *tenth-of-a-second rule*, which is intended to imply the transient nature of application code processing rather than a precise measure.
- If a message thread communicates with another thread or the operating system, then this should be done either asynchronously or so as not to violate the tenth-of-a-second rule.

---

## PM Message Queues

PM messages are generated either as the result of user interaction with the system or by the use of various PM APIs. Both PM and non-PM applications may generate PM messages.

Messages may flow:

- synchronously, that is, require processing by the recipient before the sender can continue, or
- asynchronously, that is, where no response is required.

They may flow between threads (inter-thread messages) or from a thread to itself (intra-thread messages).

These characteristics require a message queuing mechanism to be implemented so that message order may be preserved.

### Note:

A message's meaning may often depend on the outcome of a preceding message. For example, consider the action of the F4 key after the Alt key has been pressed.

Each PM message thread has two *queues* or more strictly speaking a message queue and a message list. There may be only one instance of these two structures per thread.

- The message queue is a circular array, the size of which is specified or defaulted by the application when it creates the queue using **WinCreateMsgQueue**.

This queue is used for the receipt of asynchronous messages generated by use of the **WinPostMsg** API.

- The message list has no depth and is created implicitly by **WinCreateMsgQueue**.

This is used for the receipt of synchronous messages sent using **WinSendMsg**.

**WinCreateMsgQueue** also creates a message event semaphore that is posted whenever a message, synchronous or asynchronous, is posted; or a message response is generated. This is the semaphore on which **WinGetMsg** waits for message notification.

There is a system queue, which is also a circular array. Messages are enqueued on the system queue by the **PMDD.SYS** device driver as the result of external events deriving directly from:

- Mouse activity
- Keyboard activity
- Use of a light pen
- Timer ticks.

PM maintains knowledge of who the current, mouse, keyboard, pen, and event receivers are. When an external event causes a message to be queued on the system queue, PM posts the message event semaphore of the current receiver of that particular event.

An application may define *Window Procedures* - entry points within the message thread. These are associated with a PM Window and a message queue. They receive control when a message is dispatched, that is dequeued from the message queue or message list. More than one window procedure may be serviced by the same message queue/list. Which one should be dispatched is determined from the **HWND**, which is one of the parameters associated with a message.

When **WinGetMsg** receives a message event notification, it first checks for the presence of received synchronous messages, if there are any dispatches them directly. Next it looks for an application generated (posted) message and finally for a system queue message.

The application thread explicitly dispatches asynchronous messages using **WinDispatchMsg**.

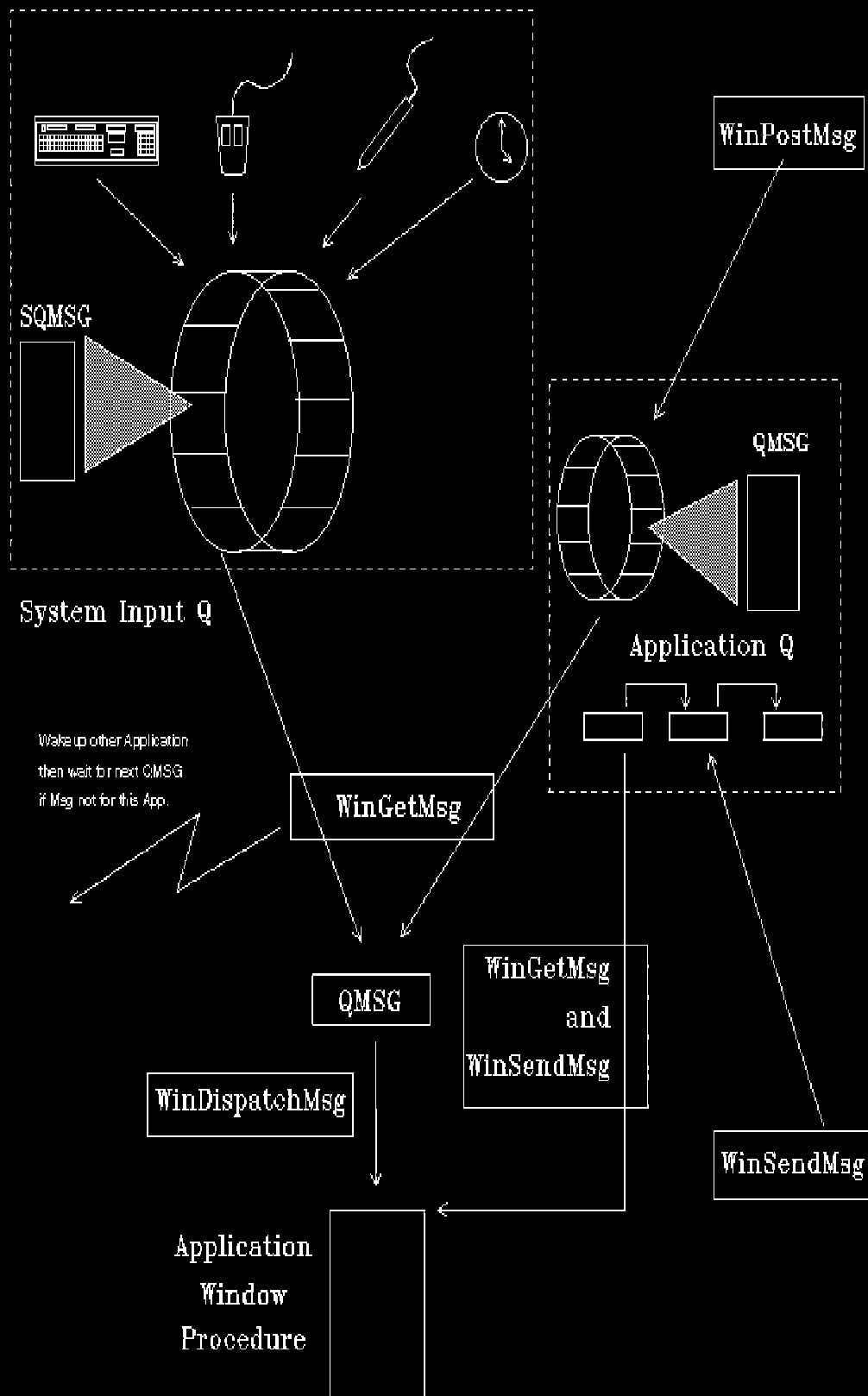
The System Queue entries are **SQMSG** structures.

The Application Queue entries are **QMSG** structures.

The Application Send Message List comprises a chain of **SMS** structures.

This scenario is illustrated in the following diagram:

## PM System Input Q Processing Overview



---

# An Application Thread's Messaging Structures

The previous section introduced the notion of a message queue and list, of where there is one pair per PM message thread. We now look at these in more detail, with the associated PM system structures that comprise the applications's messaging environment.

## The Message Queue Header (MQ)

This structure acts an anchor for all the message processing structures of an application message thread. It is created by **WinCreateMsgQueue**, and the returned **HMQ** (message queue handle) is the address of this structure. PM often refers to the address of a **MQ** as its **PMQ**.

The principle fields of interest are:

<i>Offset</i>	<i>Description</i>
+0x14	The current read position of the message queue.  Since the message queue is a circular array, four pointers have to be maintained: the current read position, current write position, top and bottom of array.  Each queue entry is a <b>QMSG</b> structure.  Entries are added to the queue in increasing address order, until the maximum (bottom) is reached then entries are added from the top.  Removal of entries only involves updating the current read pointer. Thus, a small trace of past message activity may be seen by scanning backwards from the current read pointer to the current write pointer.
+0x18	The current write pointer.
+0x24	The <b>Pid</b> of the message thread to which this <b>MQ</b> belongs.
+0x28	The <b>Tid</b> of the message thread to which this <b>MQ</b> belongs.
+0x44	The most recent <b>SMS</b> on which a response is awaited.  The presence of a non-zero value in this field implies that the message thread is currently blocked in <b>WinSendMsg</b> waiting for a response.  If the message thread recurses, for example through the receipt of a synchronous message, then a subsequent <b>WinSendMsg</b> will cause this field to be updated. The previous contents are saved on the stack.  A non-zero value in this field is of prime interest when diagnosing hangs. It immediately focuses our attention on the recipient of this message.
+0x48	The current <b>SMS</b> received.  This field is non-zero when an <b>SMS</b> is removed from the receive list for processing by its associated window procedure.  When this field is non-zero, it implies that the thread's window procedure has been dispatched to process a received message.
+09c	The Received message list.  <b>SMSs</b> are chained from this location pending dispatch.  Upon dispatch the oldest message is removed from the list and pointed to from offset +0x48 of the <b>MQ</b> .
+0xa4	The <b>thread slot number</b> of the message thread to which this <b>MQ</b> belongs.  This is very useful for correlating <b>MQs</b> to threads.

### The Send Message Structure (SMS)

The **SMS** is created for synchronous messages and linked to the receive list (**MQ+0x9c**) when **WinSendMsg** is called.

The principle fields of interest are:

<i>Offset</i>	<i>Description</i>
+0c	Pointer to the next (more recent) <b>SMS</b> in the receive list.
+14	Pointer to the <b>MQ</b> to which this <b>SMS</b> has been sent.
+18	Pointer to the <b>MQ</b> of the thread from which this <b>SMS</b> was sent.
+24	Pointer to the <b>WND</b> the represents the Window to which this message has been sent.
+28	The message Id.
+2c	Message parameter 1.
+30	Message parameter 2.

Offsets +0x14 and +0x18 are of prime interest in diagnosing hangs. They enable us to locate the recipient of a message, of which a response is pending and therefore the thread which is causing our thread to remain blocked.

### The Queue Message Structure (QMSG)

This is the structure used by applications when calling **WinDispatchMsg**.

The **QMSG** is also the form of an entry on the application's message queue.

The principle fields of interest are:

<i>Offset</i>	<i>Description</i>
+0	The window handle ( <b>HWND</b> ). This is an index (ignoring the high-order bit) in to the handle table. From the handle table we can obtain the equivalent <b>PWND</b> or pointer to the <b>WND</b> .
+4	The message Id.
+8	Message parameter 1.
+c	Message parameter 2.

### The Handle Table

This is a global table that is used to correlate window handles (**HWNDs**) with pointers to **WNDs** (**PWNDs**). The table comprises a 0x20 byte header with 8-byte entries. The first word of each entry is a **PWND** and the second a Boolean flag, which if non-zero, indicates that an **HWND/PWND** combination is non-deleteable.

The handle table may be located from the address at symbol:

pHandleTable

### The Window Structure (WND)

This structure represents a window. It is created by **WinCreateWindow**.

The **WND** has two main functions:

- >> It acts as the link between the **MQ** and the thread's window procedure
- >> It establishes the **WND** hierarchy.

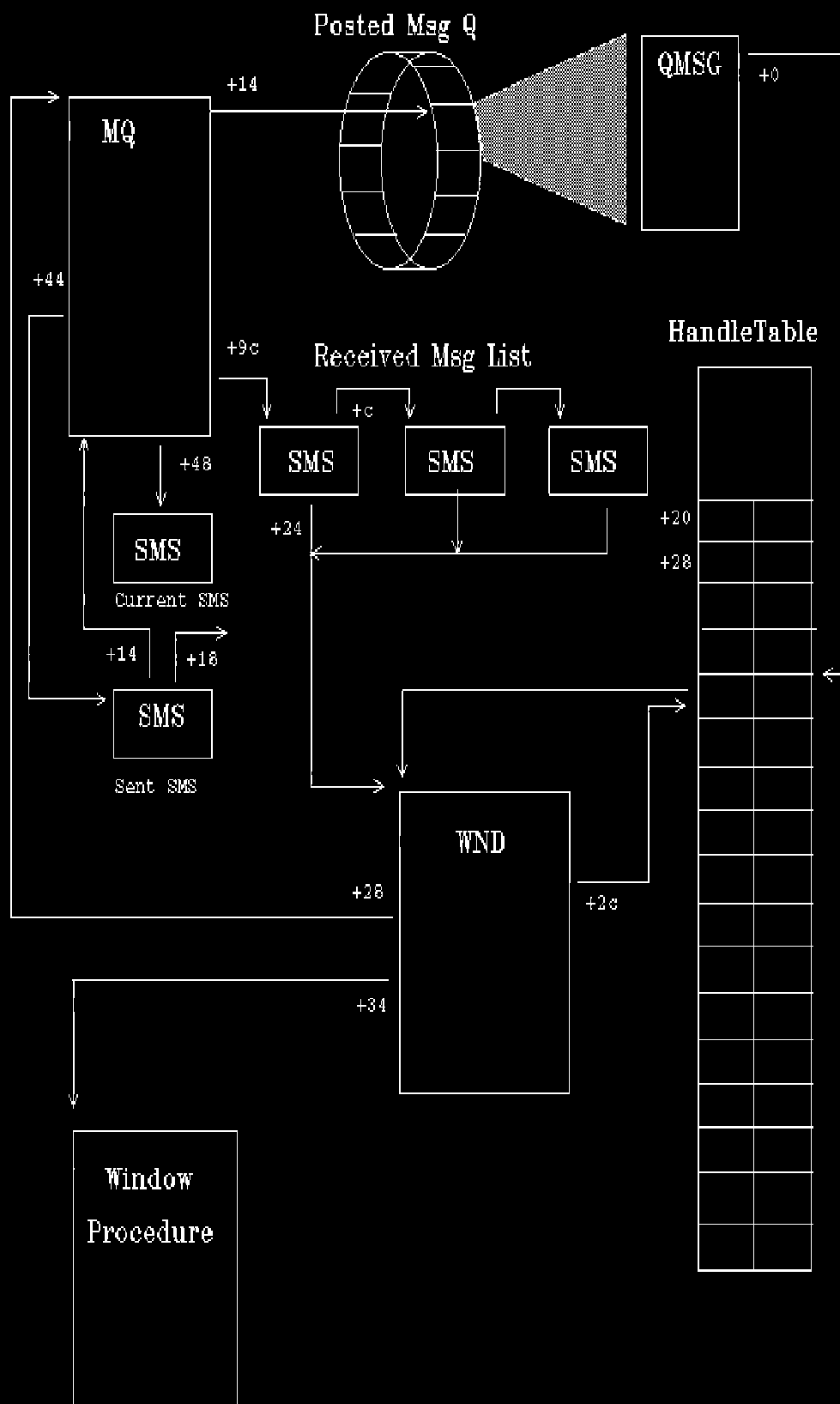
The principle fields of interest are:

<i>Offset</i>	<i>Description</i>
+0	Next sibling <b>WND</b> .
+4	Parent <b>WND</b> .
+8	First Child <b>WND</b> .

+28	The pointer to the <b>MQ</b> that will queue messages sent and posted to this window.
+2c	The windows's <b>HWND</b> .
+30	A Boolean, which if non-zero, indicates that the window procedure address is a 16-bit far pointer.
+34	The address of the window procedure.

This scenario is illustrated in the following diagram:

# PM Application Message Processing Overview





---

# PM Message Processing Logic

The following sections provide a summary of the essential internal logic of PM's message handling. This is provided to give the reader sufficient understanding that will enable most application problems, especially those that cause hangs, to be identified. In most cases hangs in the PM environment are caused by a misuse or misunderstanding of the message thread model, especially the way in which message threads act in a co-operative manner.

An outline is given for each of the following:

**WinGetMsg** logic.

**WinSendMessage** logic.

Waiting for Message Activity.

**WinSetFocus** logic.

## Note:

In each of these outlines, the added complication of calling hooks has been omitted.

---

## WinGetMsg Logic

**WinGetMsg** operates essentially as a loop that waits for message activity and returns messages to the user. Conceptually the application's code acts as an inverted subroutine of **WinGetMsg**. This was illustrated in [The PM Messaging Environment](#).

These are the essential steps in **WinGetMsg** processing:

- When **WinGetMsg** wakes, it first unlocks the System Queue if owned or if it is the *active thread*.  
  
The **MQ** of the current system queue owner is pointed to by **pmqsyslock**. This is set to zero if this points to the **MQ** of the current thread.  
  
The **active thread** is the thread that has the right to unlock the system queue if locked by another thread. Normally this is the thread that manages the **MQ** of the window in focus. Normally the thread that has locked the system queue is the **active thread**.
- The received list is checked.  
  
If **SMSs** are queued then each is removed successively and the corresponding window procedure is called.  
  
Received messages, that is messages sent via **WinSendMessage** to this thread, are not returned by **WinGetMsg**. The window procedure is called directly.
- The application's queue is checked for posted messages.  
  
If one is found it is dequeued and returned to the application.
- If no posted message is found then **WinGetMsg** tries to process the system queue.
- We attempt to lock the system queue if free.  
  
if **pmqsyslock** is zero it is set to the current thread's **MQ** address.
- If the lock was successful then we peek the next system queue message.  
  
If the lock was unsuccessful we return to the beginning of the loop and wait on the message queue semaphore.

- If the next system message is for this thread then it is dequeued and returned to the application **with the system queue lock still held**.

The possibility of holding the system queue lock while running in user code is vital to note. While this happens only **active thread** can dequeue a system queue message. The reason for holding the system queue lock is for performance. It is likely that one system message will be followed by a sequence for the same thread. If the lock was released, unnecessary processing on other message queue threads would take place. More recently queued messages could not be processed out of turn anyway, since the interpretation of a system message depends upon the outcome of the preceding system message.

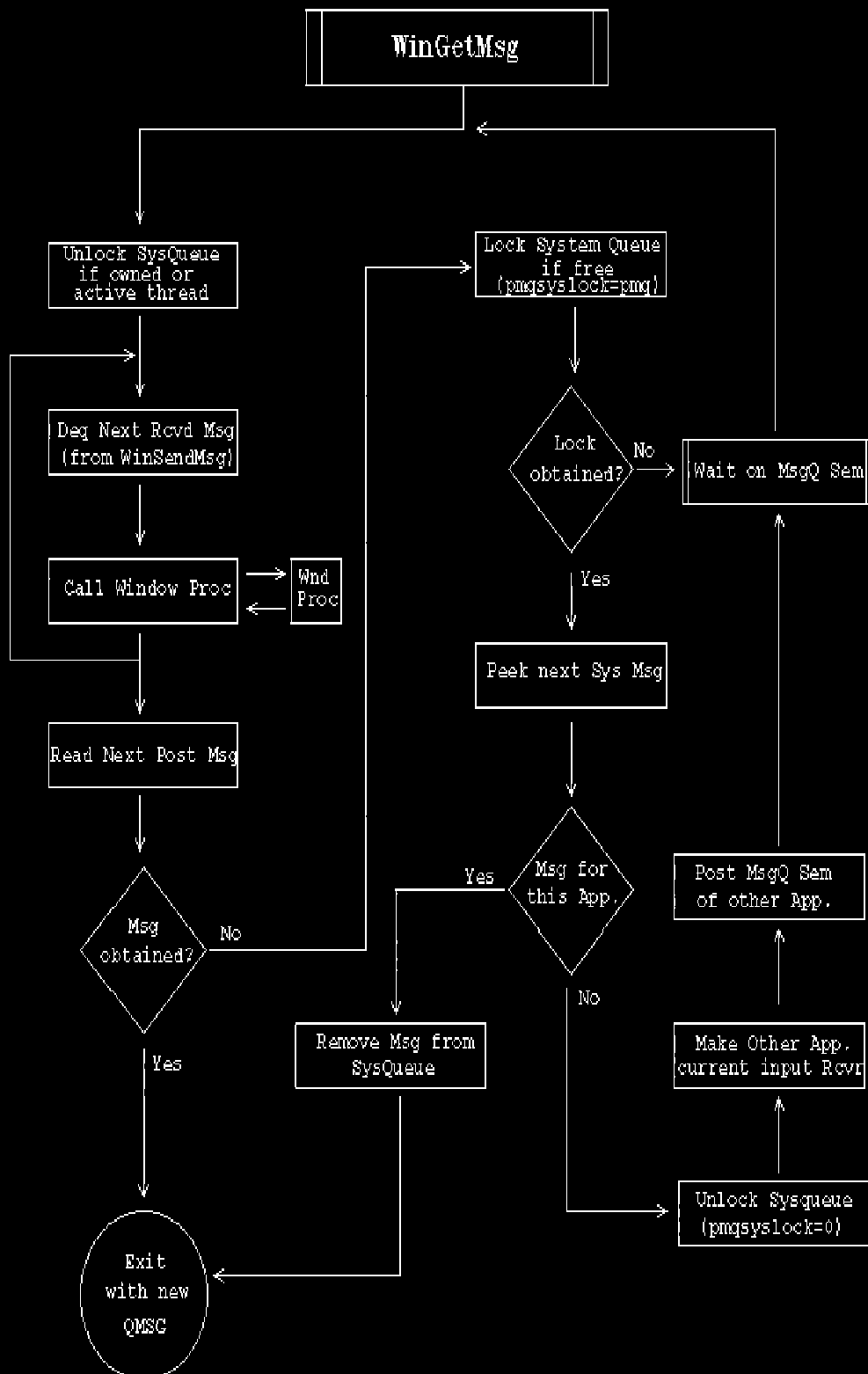
- If the next system message is for another application then the system queue is unlocked.
- The other application is made the current input receiver.

PM distinguishes between current mouse, keyboard and event receiver. **WinGetMsg** makes the other application the current mouse, keyboard or event receiver depending upon the message category.

- Finally the other application's message queue semaphore is posted. **WinGetMsg** returns to the beginning of its message loop by waiting on its own message semaphore for more message activity.

This processing is illustrated in the following diagram:

# WinGetMsg Essential Processing



---

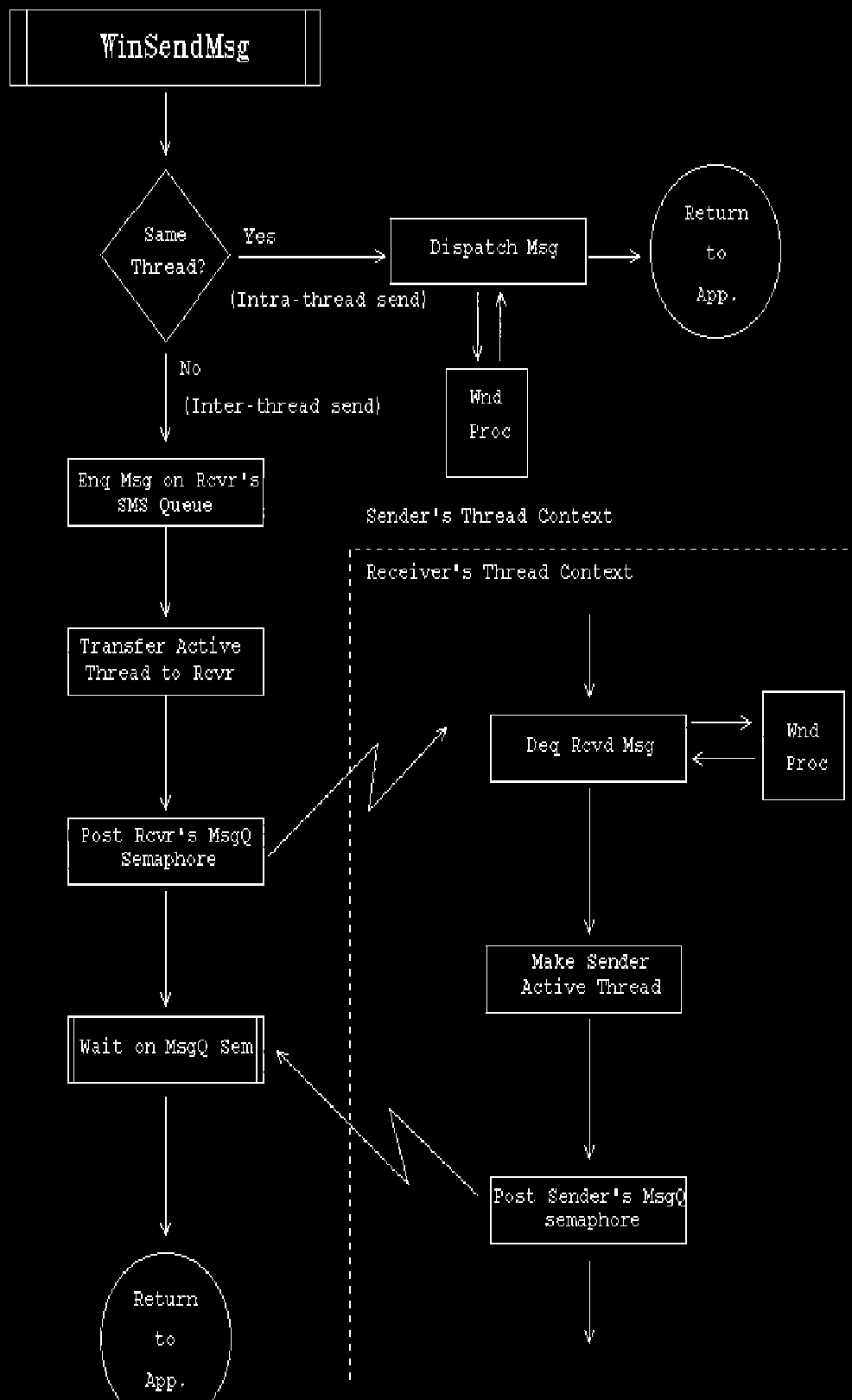
## WinSendMsg Logic

These are the essential steps in **WinSendMsg** processing:

- We check to see if the message is being sent to the same thread.  
**WinSendMsg** to the same thread is known as an *intra-thread* send.  
**WinSendMsg** to another thread is known as an *inter-thread* send.
- If intra-thread then the message is dispatched immediately, from within **WinSendMsg**.  
This behaviour implies that a window procedure may recurse many times, even if waiting for a response to a **WinSendMsg**.
- If inter-thread then the message is enqueued to the receive list of the recipient's **MQ**.
- Active thread status is transferred to the receiving thread (if currently owned).  
This allows the receiver to unlock the system queue if it had been locked by the current thread *and* the current thread is the active thread.
- The receiver's message queue semaphore is posted.
- **WinSendMsg** waits on the current thread's message queue for a response to the sent message.

This processing is illustrated in the following diagram:

# WinSendMsg Essential Processing



---

## Waiting for Message Activity

In order to process synchronous messages as swiftly as possible, PM always checks the receive list of the current thread for pending **SMSs** before waiting on an internal PM semaphore.

If **SMSs** are found queued, they are successively dispatched.

Only when the receive list has been processed does PM finally wait on a semaphore.

This applies particularly to the message processing semaphore, but also equally to semaphores that serialise access to resources such as the .INI files.

---

## WinSetFocus Logic

**WinSetFocus** has a subtle bearing on message processing since it selects a new active thread and new current input receivers for system messages.

**Note:**

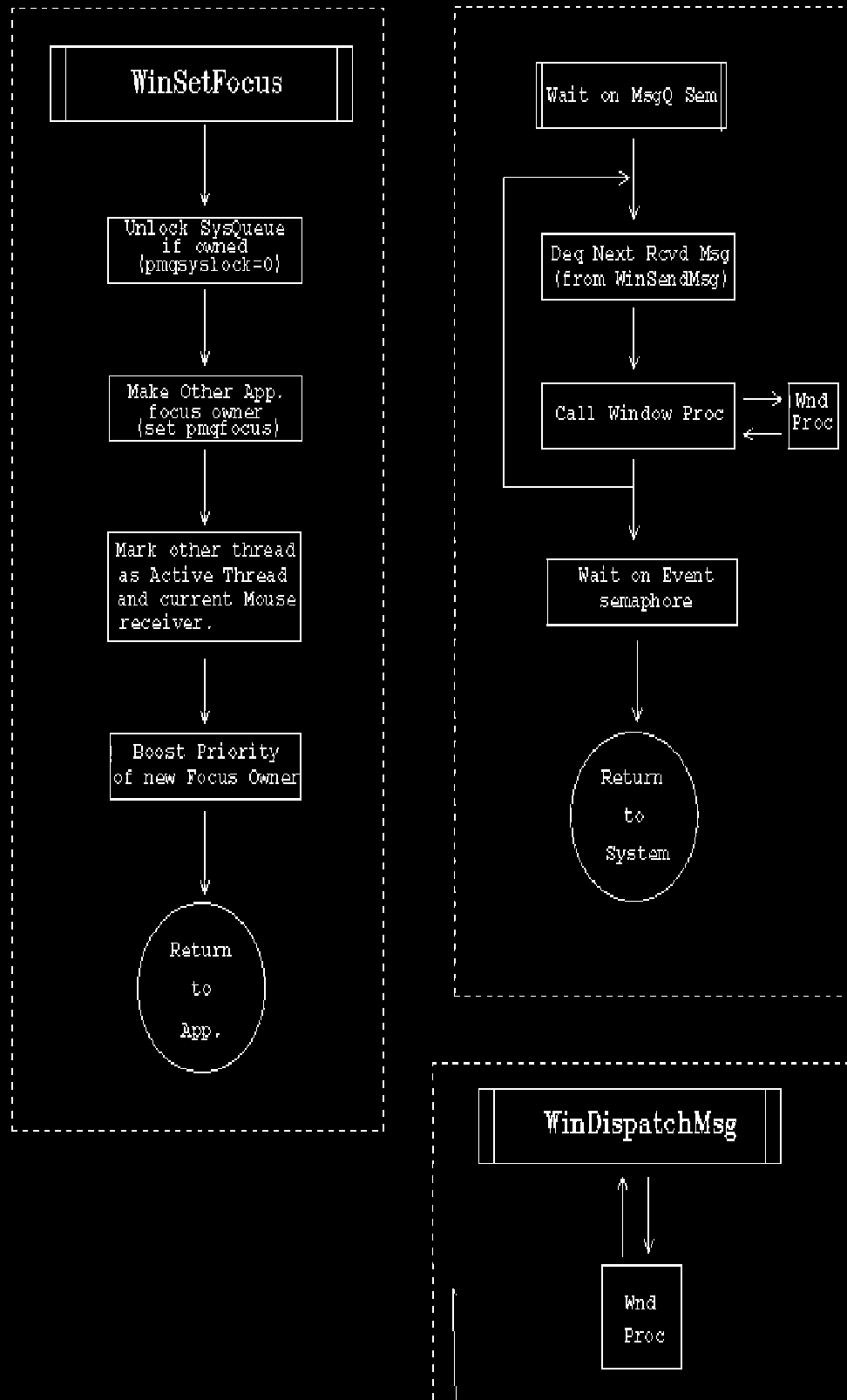
Focus may be changed by a third party.

These are the essential steps in **WinSetFocus** processing:

- **WM\_FOCUSCHANGE** is sent to the message thread losing the focus.
- The current window in focus is changed.  
**pwndfocus** points to the **WND** of the focus owner.
- Unlock the system queue if locked.  
The **MQ** of the current system queue owner is pointed to by **pmqsyslock**. This is set to zero if it points to the **MQ** of the current focus owner.
- The target window's message thread is marked as the new focus owner.  
**pmqfocus** is set to the address of the new focus owner's **MQ**.
- The target thread's message queue is made the current mouse and keyboard input receiver.  
**pmqMouseWake** and **pmqKeyWake** are set to the address of the new focus owner's **MQ**.
- The new focus owner's message thread is marked as the new active thread.
- A priority boost is applied to the new focus owner's message thread.
- **WM\_FOCUSCHANGE** is sent to the message thread gaining the focus.

This processing is illustrated in the following diagram:

## Miscellaneous PM Processing



---

# Application Not Responding to Messages Logic

The **Application Not Responding to Messages** dialog, or **BadApp** dialog as it is sometimes referred to, appears after Ctrl-Esc has been hit and the system has not been able to display the task list.

The essential logic for this processing is as follows:

- Ctrl-Esc is hit and 5 second timer is started.
- If the task list appears before the time-out then all is well, if not we check for who might be holding things up.
- We try for 5 seconds to obtain the User PM Semaphore. If unsuccessful then the Pid, Tid and Session Id of the owner is saved in the **QHPSTRUCT**.
- If **pmqsyslock** is owned then the Pid, Tid and Session Id of the owner is saved in the **QHPSTRUCT**.
- If **pmqfocus** is owned then the Pid, Tid and Session Id of the owner is saved in the **QHPSTRUCT**.
- We now enter a second wait of a further 3 seconds, after which, we build a second **QHPSTRUCT**.
- If the Tid and Pid are different and we have not yet had an acknowledgement from the task list then we wait a further 12 seconds, on the assumption the processing is slow, but not hung.
- If the Tid and Pid are the same or we have expired on our third time-out then we set **fBadAppDialog** true and reset the cause for the hang:
  - If **User\_Sem** held then release **User\_Sem**
  - if system queue locked then reset **pmqsyslock**
  - if focus owner hung, then reset **pmqfocus**

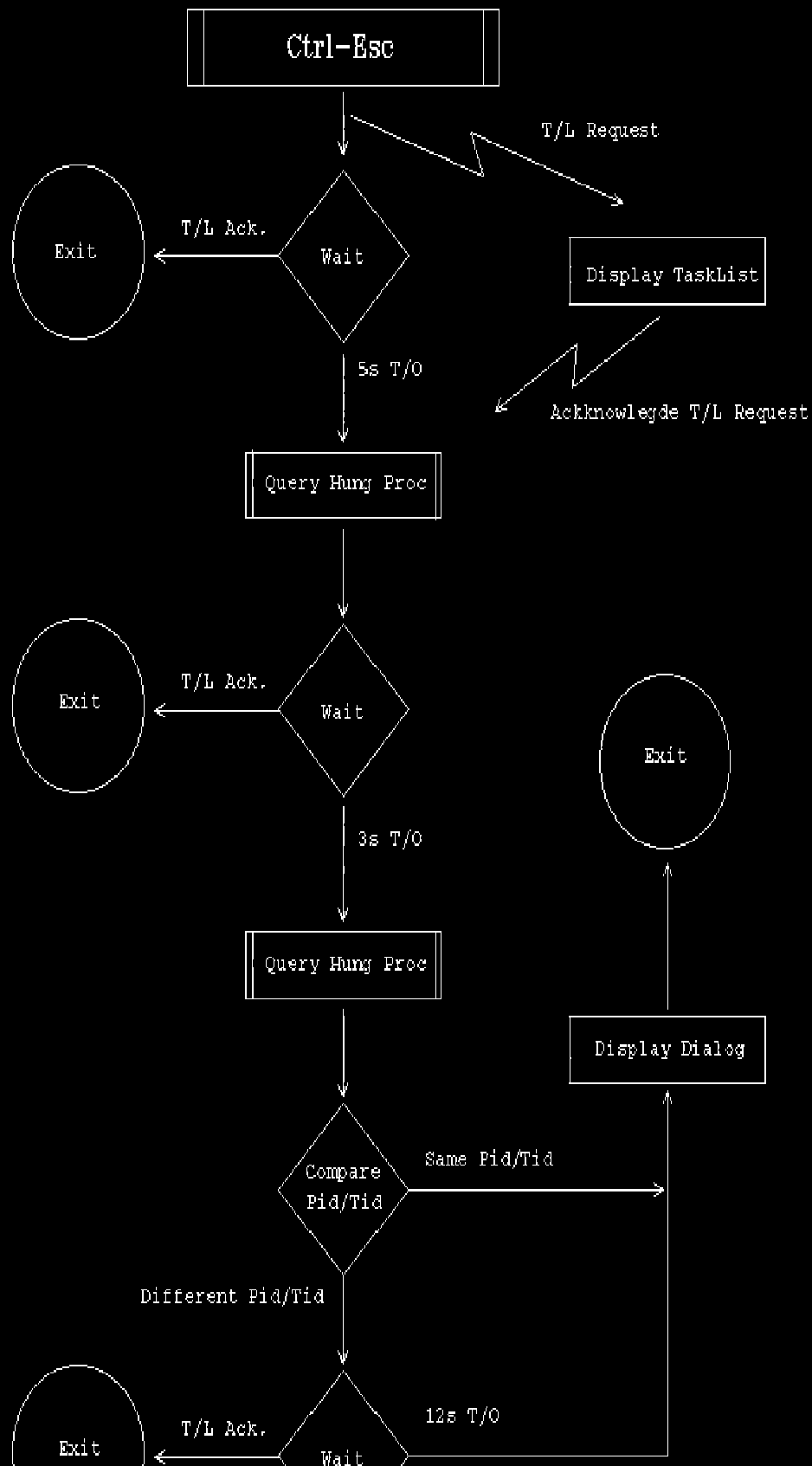
Report the hanging application in the **BadApp** dialog.

This processing is illustrated in the following two diagrams:

## BadApp Dialog Logic

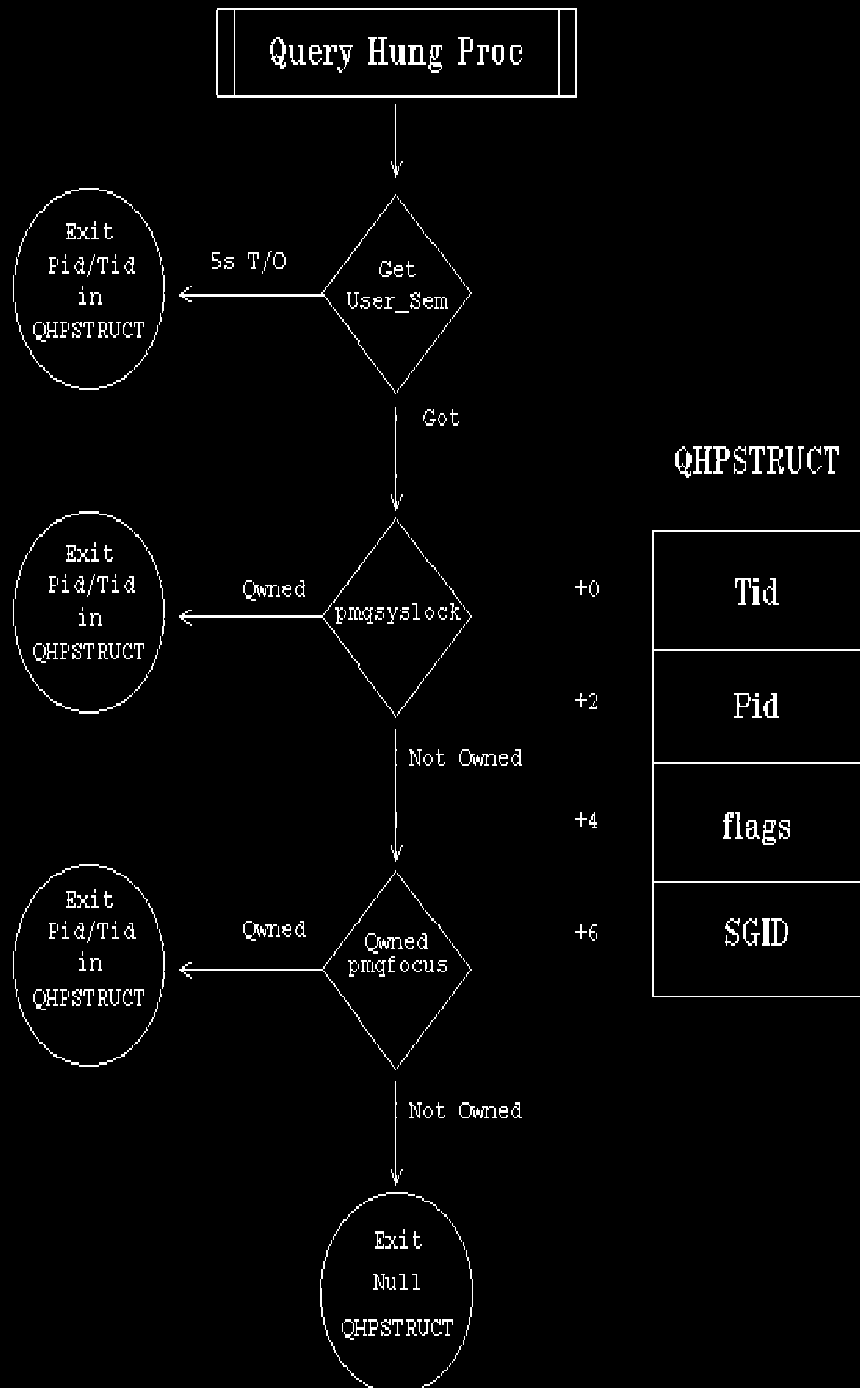


# BadApp Dialog Processing



Query Hung Process Logic

## Query Hung Process Processing



The flags in the **QHPSTRUCT** indicate the detected reason for hanging. These may be a combination of:

Name	Bit Mask	Description
QHP_SYSQUEUELOCK	0x0001	System Queue Locked
QHP_SENDSMSGLOCK	0x0002	Waiting for a response to WinSendMsg
QHP_CLIPBRDLOCK	0x0004	
QHP_WINDOWLOCKED	0x0008	
QHP_VISRGNLOCKED	0x0010	
QHP_LOCKWINDOWUPDATE	0x0020	
QHP_FSRUSERHANG	0x4000	Waiting for the User_Sem
QHP_INPUTPROCESSED	0x8000	

-----

## Useful Symbols for PM Structures

The following list is a small selection of global symbols from **PMMERGE.SYM** that will be of help in locating the structures associated with message handling:

### pmsemaphores

This is the label for the table of [PMSEM](#) and [GRESEM](#) semaphore structures. Offset +0x20 is the location of the User Semaphore. If this is owned and not released within the time-out period after Ctrl-Esc has been hit, then the **Application Not Responding to Messages** reports the semaphore owner as the culprit.

### pmqSyslock

The **PMQ** of the thread that has locked the system queue.

This is the first place to look when investigating a hang in a PM application. The system uses this to name a *bad* application when the **Application Not Responding to Messages** dialog appears.

### pmqFocus

The **PMQ** of the window that has the focus.

if **pmqSyslock** is zero, the system uses this as a second choice for the *bad* application when the **Application Not Responding to Messages** dialog appears.

### pmqKeyWake

The **PMQ** of the current keyboard event receiver.

### pmqMouseWake

The **PMQ** of the current mouse event receiver.

### pmqEventWake

The **PMQ** of the current miscellaneous event receiver.

### pwndFocus

The **PWND** of the window currently in focus.

### pmqShutdown

The **PMQ** of the thread sent a **WM\_QUIT** and being waited on to terminate. If Shutdown doesn't complete, then this could be a good place to start investigation of the problem.

### pmqCapture

The **PMQ** of the thread that has the mouse captured.

### pwndCapture

The **PWND** of the window associated with mouse capture.

<b>pwndSysModal</b>	The <b>PWND</b> of the System Modal window.
<b>pmqVisLock</b>	The <b>PMQ</b> that has visible regions locked.
<b>pmqTrack</b>	The <b>PMQ</b> of the thread currently in <b>WinTrackRect</b> .
<b>pmqLockUpdate</b>	The <b>PMQ</b> of the thread that has update locked.
<b>pHandleTable</b>	The address of the handle table.
<b>pSysqueue</b>	The <b>MQ</b> of the system input queue.  The system queue is headed by a partial <b>MQ</b> since it does not require the fields to support a receive list.
<b>pmqShell</b>	The <b>PMQ</b> of the 1st thread of the 1st Shell Process.  This thread is responsible for starting and re-starting the Workplace Shell.
<b>pmqShell2</b>	The <b>PMQ</b> of the 1st thread of the 2nd Shell or Workplace Shell Process.  This thread is the main thread of the desktop PM application.
<b>paAABRegs</b>	The address of the application anchor block registers ( <b>AAB</b> ).  <b>AAB</b> registers are allocated for each PM application message thread. This is located in thread local memory, which implies that it is correct only for the current thread context. The Thread Local Memory Area is saved in the <b>TCB</b> and restored when the thread is made current. Since the <b>TCB</b> may be located in System storage under any context then a thread's <b>PMQ</b> may be found in any context from its <b>TCB</b> .
<b>pwndObject</b>	The <b>PWND</b> of the Object Window.  This is the parent or owner of all non-display windows.
<b>pwndDesktop</b>	The <b>PWND</b> of the Desktop Window.  This is the parent of owner of all displayable windows.
<b>SleepPMQ+nnn</b>	When a thread is blocked at approximately offset +0x155 into <b>SleepPMQ</b> then it is waiting on the message queue semaphore for new messages or responses to outstanding sent messages. Offset +0x30 from the current stack pointer usually contains the <b>PMQ</b> for the current thread.
<b>pmqList</b>	All <b>MQs</b> are chained on a single linked master list from offset +0x0 of the MQ. The current head of the master list is pointed to by <b>pmqList</b> .
<b>psmsList</b>	All <b>SMSs</b> are chained on a single linked master list from offset +0x0 of the <b>SMS</b> . The current head of the master list is pointed to by <b>psmsList</b> .
<b>qhpsBadApp</b>	This is the label of the <b>QHPSTRUCT</b> saved the first time a time-out occurs after Ctrl-Esc has been hit and the <b>Application Not Responding to Messages</b> dialog appears.
<b>fBadAppDialog</b>	A Boolean that indicates when the <b>Application Not Responding to Messages</b> dialog has been displayed.

-----

## Useful PM Structures

The following diagrams illustrate the main PM structures for the messaging function. These are laid out as if viewed under the Kernel Debugger or Dump Formatter by displaying them using the [DD Command](#).

- [PM Message Queue Header \(MQ\)](#)
- [PM Window Structure \(WND\)](#)
- [PM Message Structures \(SMS, QMSG, SQMSG\)](#)
- [PM Application Anchor Block Registers](#)
- [Stack Layout at Useful Entry Points](#)

---

# PM Message Queue Header

# PM Message Queue Header

## viewed as double-words

+0	Next MQ in master list	Number queued	Q entry length		Queue Depth	Top of Queue
+10	Bottom of Queue	Next QMSG to Read		Next QMSG to Write		
+20		PID		TID		SCID
+30	MSQ Event Sem handle					
+40		Current Sent SMS (WinSendMsg)		Current Rcvd SMS (WinSendMsg)		
+90						Rcvd SMS List pending dispatch
+a0		Thread Slot				

---

## PM Window Structure (WND)



PM Window Structure (WND)  
viewed as double-words

+0	Next Sibling WND	Parent WND	Child WND	Owner WND
+10				User specified id
+20			PMQ	HWND
+30	16-bit BOOL	Window Proc address		
+40				
+50				
+60	Window words start here			

-----

# PM Message Structures (SMS, QMSG, SQMSG)

## PM Send Message Structure (SMS) viewed as double-words

+0	Next SMS on master list	Send List Head		Next SMS in Receive List
+10	time	Sender PMQ	Receiver PMQ	Result
+20		PMND	Msg Id	MP1
+30	MP2			

## PM Queue Message Structure (QMSG) viewed as double-words

+0	HMND	Posted Msg Id	MP1	MP2
+10	time	X Co-ord	Y Co-ord	

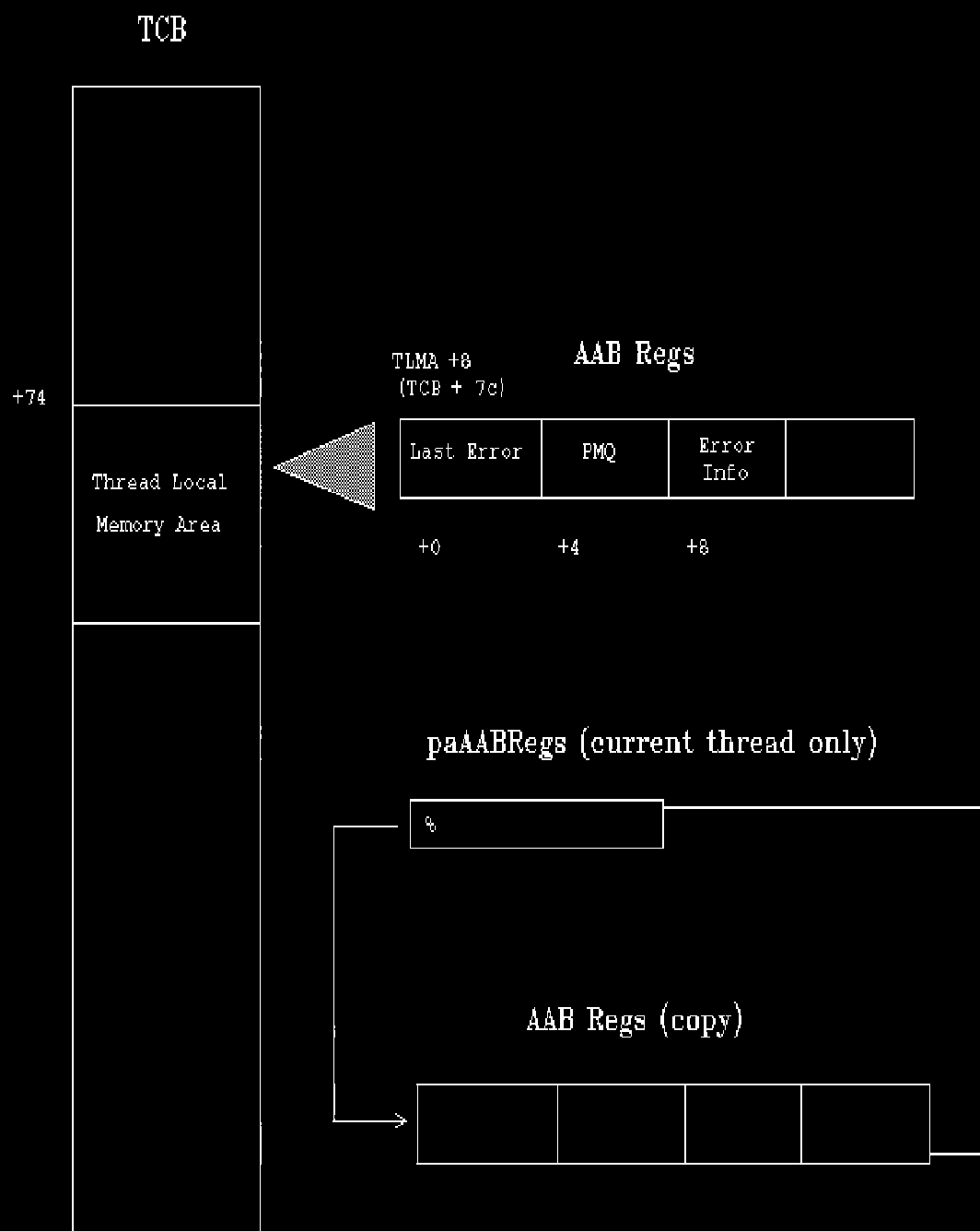
## PM System Queue Msg Structure (SQMSG) viewed as double-words

+0	Msg Id	MP1	MP2	time
+10				

-----

# PM Application Anchor Block Registers

# PM Application Anchor Block



---

## Stack Layout at Useful Entry Points

## Stack Frames for Common Entry Points viewed as double-words

Win32PostMsg Entry Point

Win32SendMsg Entry Point

Window Procedure Entry Point

+0	Next Frame pointer	Return Address following call to entry point	HWND	Msg ID
+10	MP1	MP2		

Win32DispatchMsg Entry Point

+0	Next Frame pointer	Return Address following call to entry point	HAB	Pointer to QMSG
+10				

---

# PM Worked Examples Under WARP

We give two examples of diagnosing common application problems:

[Example 1](#) - A trap in PMMERGE.DLL caused by an application fault.

[Example 2](#) - A hang in the WorkPlace, again caused by an application fault.

Further techniques are illustrated in:

[How to find the MQ of any thread.](#)

[How to find the MQ of a BadApp Application.](#)

[Finding Application and System Queue Elements.](#)

Further examples, with annotated solutions, may be found on the accompanying CD-ROM in the **TURKEY** lab exercise.

---

## Example 1 - A Trap in PMMERGE.DLL

### Steps for analysing traps in PM DLLs:

- Intercept the trap at the point of failure.
- Unwind the stack to the application call.
- Validate the parameters to the API call.
- If necessary, determine how the user routine was invoked by examining the **MQ** and looking for dispatched messages or by unwinding the stack further.

This example is of a trap in PMMERGE.DLL, but caused by an application fault!

Because we have a trap E, we set the fatal vector under the Kernel Debugger (or use TRAPDUMP=ON in CONFIG.SYS to take a dump) then re-create the problem.

```
##vsf *
##g
Trap 14 (0EH) - Page Fault 0006, Not Present, Write Access, User Mode
eax=00000007 ebx=00273fcc ecx=00000001 edx=00000007 esi=12d3e089 edi=00000000
eip=1bd3d261 esp=00273ebc ebp=00273f1c iopl=2 rf -- -- nv up ei pl nz ac po cy
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001d4000
005b:1bd3d261 f3a5             repe movsd es:00000000=invalid ds:12d3e089=69727453
##ln
005b:00000000 turkey:FLAT:__$dummy$ + 1bd3d261
```

We have trapped in a DLL, probably PMMERGE.DLL, certainly not in the user's .EXE code. We unwind the stack to find the return address in the user's .EXE

```
##dd %ebp
%00273f1c 00273f5c 1bd3d05b 00000000 00000005
%00273f2c 00080001 00000000 00080013 00010423
%00273f3c 00190008 00000000 00000001 00000014
%00273f4c 00000000 80000144 00000001 00190008
%00273f5c 00273fa8 1bd03893 80000144 00000071
%00273f6c 00280018 00000000 000103ec 00000000
%00273f7c 00000000 00190008 00273fcc 00000000
%00273f8c 00000000 ffffffff 80000144 12d31630
##d
%00273f9c 00000000 12d31494 00190008 00273ff4
%00273fac 00010932 00190008 00273fcc 0000000c
```



```

%00273fbc  00000004 00000004 00000007 80000144
%00273fcc  80000144 00000071 00280018 00000000
%00273fdc  0092d87d 0000027a 000000f0 00000000
%00273fec  12d31630 00190008 00000000 1bfbbf68
%00273ffc  00022bf4
Invalid linear address: %00274000

```

Not all the stack was paged in to physical memory, but never mind. Enough is there to allow us to find the return address to the user's application code.

Following the base pointer (EBP):

```

%00273f1c  00273f5c 1bd3d05b

%00273f5c  00273fa8 1bd03893 80000144 00000071

%00273f9c                                00273ff4
%00273fac  00010932 00190008 00273fcc 0000000c

```

The return address is %10932. We inspect the code just before this address.

```

##u %10932-20
%00010912 25837ddc2a      and     eax,2adc7d83
%00010917 0f8507000000    jnz     %00010924
%0001091d e916000000    jmp     %00010938
%00010922 8bc0      mov     eax,eax
%00010924 8d45d8      lea     eax,[ebp-28]
%00010927 50      push    eax
%00010928 ff75fc      push    dword ptr [ebp-04]
%0001092b b002      mov     al,02
%0001092d e8862ecf1b    call    %1bd037b8
%00010932 83c408      add     esp,+08
%00010935 ebc1      jmp     %000108f8
%00010937 fc      cld
##ln %1bd037b8
%00000000 turkey:FLAT:__$dummy$ + 1bd037b8

```

This call was to a routine at %1bd037b8. LN doesn't give us a useful symbol (our .EXE is being selected instead of the correct .DLL symbol). We find out who owns this address from the memory management control blocks.

```

##.m %1bd037b8

*har      par      cpg      va      flg next prev link hash hob      hal
01c0 %feaf168a 00000001 %ffe3c000 101 000d 01e2 01bd 0000 013e 000c      =0000
hal=000c pal=%ffe5c078 har=01c0 hptda=010c pgoff=000d7 f=021
har      par      cpg      va      flg next prev link hash hob      hal
01bd %feaf1648 00000001 %ffede000 101 01a2 01be 0106 0000 013e 0009      =0000
hal=0009 pal=%ffe5c060 har=01bd hptda=010c pgoff=00000 f=021
har      par      cpg      va      flg next prev link hash hob      hal
0106 %feaf068e 000000f0 %1bd00000 3d9 0105 0107 0000 0000 013e 0000 hco=00307
hob      har hobnxt flgs own hmtc sown,cnt lt st xf
013e 01c0 0000 1838 0139 0139 0000 00 02 00 00 shared      c:pmmerge.dll
hco=00307 pco=fe64ef3e hconext=00296 hptda=032f f=1c pid=0019 a:turkey.exe
hco=00296 pco=fe64ed09 hconext=001f9 hptda=0301 f=1c pid=0009 d:cmd.exe
hco=001f9 pco=fe64e9f8 hconext=00188 hptda=02c9 f=1c pid=0007 d:cmd.exe
hco=00188 pco=fe64e7c3 hconext=00107 hptda=0291 f=1c pid=0005 c:2000.exe
hco=00107 pco=fe64e53e hconext=0008e hptda=023b f=1c pid=0004 c:pmshell.exe
hco=0008e pco=fe64e2e1 hconext=0002a hptda=01b5 f=1c pid=0003 c:harderr.exe
hco=0002a pco=fe64e0ed hconext=00000 hptda=010c f=1d pid=0002 c:pmshell.exe

```

We can see that our call was to an entry point in PMMERGE.DLL. We need to activate PMMERGE's symbols.

```

##wa pmmerge
##ln %1bd037b8
%1bd037b8 pmmerge:PM32BIT:WIN32DISPATCHMSG
##ln
005b:1bd3d064 pmmerge:PM32BIT:LoadStrMsg + 1fd
005b:1bd3d2a8 WIN32POSTQUEUEMSG - 47

```

So we called WinDispatchMsg and some time later we probably called LoadStrMsg, which is where we trapped. First we need to check the

parameters to WinDispatchMsg. These are:

HAB 00190008

PQMSG 00273fcc

The QMSG at %273fcc is also in the stack we dumped:

```
%00273fcc 80000144 00000071 00280018 00000000
%00273fdc 0092d87d 0000027a 000000f0 00000000
```

The first parameter is the **HWND**. We convert this to a **PWND**, dump the **WND** and look for the window procedure entry point.

```
##dd phandletable_l1
9f3f:0000ab78 12d50000
```

```
##dd %12d50000+20+(8*144) 12
%12d50a40 12d31494 00000000
```

```
##dd %12d31494
%12d31494 12d31838 12d3c974 00000000 12d3c974
%12d314a4 00c80262 0104029e 80000000 00000008
%12d314b4 12d314f0 00000004 12d31630 80000144
%12d314c4 00000000 000103ec 00000000 00000000
%12d314d4 12d3147c 00000000 00000000 00000000
%12d314e4 00000000 2050534d 00000034 12d31894
%12d314f4 00004b4e 00000000 0000fc4e 00000019
%12d31504 00000000 000103ec 00000000 00000000
```

#### Note:

We could have used the following more complex single command construct to achieve the same result:

```
##dd %(dw(%(dw(phandletable))+20+(8*144)))
%12d31494 12d31838 12d3c974 00000000 12d3c974
%12d314a4 00c80262 0104029e 80000000 00000008
%12d314b4 12d314f0 00000004 12d31630 80000144
%12d314c4 00000000 000103ec 00000000 00000000
%12d314d4 12d3147c 00000000 00000000 00000000
%12d314e4 00000000 2050534d 00000034 12d31894
%12d314f4 00004b4e 00000000 0000fc4e 00000019
%12d31504 00000000 000103ec 00000000 00000000
```

The window procedure entry point is at offset +0x34.

We now unassemble this:

```
##u %103ec
%000103ec 55          push    ebp
%000103ed 8bec          mov     ebp,esp
%000103ef 83ec08        sub     esp,+08
%000103f2 8b4508        mov     eax,dword ptr [ebp+08]
%000103f5 a32c0d0200   mov     dword ptr [00020d2c],eax
%000103fa 8b450c        mov     eax,dword ptr [ebp+0c]
%000103fd e93a000000   jmp     %0001043c
%00010402 8bc0        mov     eax,eax
%00010404 ff7508        push   dword ptr [ebp+08]
%00010407 b001        mov     al,01
%00010409 e8322dcf1b   call   WIN32QUERYANCHORBLOCK (%1bd03140)
%0001040e 8945fc        mov     dword ptr [ebp-04],eax
##u
%00010411 6a00          push   +00
%00010413 6a14          push   +14
%00010415 6a01          push   +01
%00010417 6a00          push   +00
%00010419 ff75fc        push   dword ptr [ebp-04]
%0001041c b005        mov     al,05
%0001041e e81dccc21b   call   WIN32LOADSTRING (%1bd3d040)
%00010423 83c418        add     esp,+18
%00010426 eble          jmp     %00010446
```

```

%00010428 8b4508      mov     eax,dword ptr [ebp+08]
%0001042b e8d0fbffff    call    main (%00010000)
%00010430 eb14        jmp     %00010446

```

We notice that we trapped in an internal routine called LoadStrMsg and that we have called WinLoadString in the window procedure. Could these be related?

We see from the PM Programming Reference that WinLoadString has 5 parameters. The right most is a pointer to a buffer and we see that the window procedure has pushed 0 on the stack this will surely cause WinLoadString to trap at some point. How do we make this supposition less circumstantial and more concrete?

Clearly, for EBP to take us back to a call to WinDispatchMsg, without finding a stack frame from the window procedure implies that PMMERGE is using optimised code when the trap occurred. That is, the conventional use of EBP is not in place - and this does occur in many internal routines in PMMERGE, for performance reasons. If we scan back through the stack we notice the address %10423 occurring shortly before (in time) the call to LoadStrMsg. This address is the return address from the WinLoadString call in the window procedure. It would seem therefore that we have called that API with the bad parameter as suspected!

## Example 2 - A Hang in a PM Application

### Steps for analysing hangs in PM applications:

- Determine whether there is a general hang in the PM environment, or a just in one application. If the latter then proceed with normal hang analysis.
- Check whether the **User\_Sem** is owned. If it is then this may be an indication of a problem. Determine the owner and their thread status.
- Check **pmqsyslock** to see if the system queue is locked. If it is, then determine the owner of the lock and their thread status.
- Check **pmqfocus** if neither of the preceding checks reveals anything informative. Determine the thread in focus and its status.
- If **pmqfocus** is a shell thread, check **fBadAppDialog**. If it is non-zero then analyse the **QHPSTRUCT** at label **qhpsbadapp**.
- If none of the preceding steps yields any results then check the shell processes. In particular **pmqshell** and **pmqshell2**. Most of the time these threads should be waiting for a message to arrive. Any other state should be transient.

This example is of a hang in the WorkPlace caused by a PM application fault.

First we check out whether the **User\_Sem** is held, whether the system queue is locked and if necessary who has the focus.

```

##db pmsemaphores+20 120
9f3f:0000b4d4 50 4d 53 45 4d 00 00 00-00 00 00 00 00 00 00 00 PMSEM.....
9f3f:0000b4e4 00 00 00 00 00 00 00 00-03 00 01 80 00 00 00 00 .....
##dd pmqsyslock 11
9f3f:0000ed14 12d3128c
##dd %12d3128c
%12d3128c 12d31630 00000020 0000000a 12d31334
%12d3129c 12d31474 12d31334 12d31334 0000a400
%12d312ac 00000000 0000001a 00000009 00000012
%12d312bc 80030059 0097ec67 00000048 00000089
%12d312cc 00000001 12d3ff5c 00000000 00000010
%12d312dc 00000000 00000000 00000000 00000000
%12d312ec 00000000 00000000 00000000 00000000
%12d312fc 00000000 00000000 8000016e 00000000
##d
%12d3130c 0fe90000 00005453 00000325 00000000
%12d3131c 12d31228 0bfff002 00000000 00000000
%12d3132c 00000001 0000002e 00000000 00000000
%12d3133c 00000000 00000000 00000000 00000000
%12d3134c 00000000 00000000 00000000 00000000
%12d3135c 00000000 00000000 00000000 00000000
%12d3136c 00000000 00000000 00000000 00000000
%12d3137c 00000000 00000000 00000000 00000000
##.p2e
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
002e 001a 0002 001a 0009 blk 0500 ab911000 ab9c9408 ab9bc6c0 led0 12 turkey
##.s 2e
##.r
eax=80030059 ebx=00008000 ecx=00090000 edx=00000004 esi=ffffffff edi=12d3128c

```

```

eip=1bd0c8e1 esp=00293ea8 ebp=00090000 iopl=2 -- -- -- nv up ei pl zr na pe nc
cs=005b ss=0053 ds=0053 es=0053 fs=150b gs=0000 cr2=12d3028c cr3=001d4000
005b:1bd0c8e1 83c40c add esp,+0c
##ln
005b:1bd0c78c pmmerge:PM32BIT:SleepPmq + 155
005b:1bd0c940 CalcWakeBits - 5f

```

No one Owns the **User\_Sem** since words at offsets +0x8 and +0xa are both zero.

We see that the system queue is held by slot 2e, who happens to be blocked in PMMERGE which is waiting for message activity. We also notice that at MQ+44 there is a non-zero value, which indicates that this thread has called WinSendMsg and is waiting for a response.

We investigate the WinSendMsg by examining the SMS pointed to by MQ+44

```

##dd %12d3ff5c
%12d3ff5c 00000000 12d3ff5c 00000000 00000000
%12d3ff6c 0097ec67 12d3128c 12d32cb4 00000000
%12d3ff7c 00000000 12d33168 00000071 00250016
%12d3ff8c 00000000 12d3ff5c 12d3ff5c 00000000
%12d3ff9c 00000000 0092e954 12d3910c 12d3ca34
%12d3ffac 00000000 00000002 12d34fac 00000407
%12d3ffbc 00000000 00000000 12d3ff90 12d3ff5c
%12d3ffcc 00000000 00000000 0092834a 12d3910c

```

The target MQ for the sent message is at offset +18, i.e. %12d32cb4

We find out who this is (the slot number is at MQ+a4).

```

##dd %12d32cb4
%12d32cb4 12d34940 00000020 0000000a 12d32d5c
%12d32cc4 12d32e9c 12d32d5c 12d32d5c 04002fff
%12d32cd4 04000400 0000001a 00000001 00000012
%12d32ce4 80030051 0093c378 00000000 00000000
%12d32cf4 00000000 00000000 00000000 00000010
%12d32d04 00000000 00000000 00000000 00000000
%12d32d14 00000000 00000000 00000000 00000000
%12d32d24 00000000 00000000 8000006c 00000000
##d
%12d32d34 0fe90000 00005453 00000325 00000000
%12d32d44 12d33304 0bfff000 00000000 12d3ff5c
%12d32d54 00000001 00000028 00000000 00000000
%12d32d64 00000000 00000000 00000000 00000000
%12d32d74 00000000 00000000 00000000 00000000
%12d32d84 00000000 00000000 00000000 00000000
%12d32d94 00000000 00000000 00000000 00000000
%12d32da4 00000000 00000000 00000000 00000000
##.p 28
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0028 001a 0002 001a 0001 crt 0500 ab905000 ab9c9408 ab9bbaf0 1f10 12 turkey

```

Offset +a4 gives us the slot number which turns out to be another thread of the turkey application. The status of this thread is **crt!** This indicates that some other thread in the same process has entered critical section, furthermore slot 28 would be ready to run had it not been for the critical section thread. Clearly this is why our application has hung the PM messaging function. The real culprit is the user of Critical Section, who is it?

The PTDA contains the address of the TCB in critical section. The TCB offset +0 contains the thread id followed by the slot number.

```

##dd %ab9c9408+ptda_ptcbcritsec-ptda_start 11
%ab9c96e8 ab9bc6c0
##dd %ab9bc6c0 11
%ab9bc6c0 002e0009
##.p 2e
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
002e# 001a 0002 001a 0009 blk 0500 ab911000 ab9c9408 ab9bc6c0 1ed0 12 turkey

```

Our application has perpetrated one if not two faults:

- First, we are using DosEnterCriticalSection in a PM application. This is a very heavy-handed way of serialising and likely to impact PM message processing, particularly if one of the other threads in the application holds the system queue lock.
- Secondly and more seriously, the thread that has entered critical section has subsequently called an API. The consequences of this are unpredictable and can lead to a hang as illustrated. Furthermore, this would apply whether or not the application was

running in a PM environment.

## How to Find the MQ of Any Thread

This example illustrates a basic technique for finding the **MQ** for a specific thread.

We find the **MQ** for thread slot 8:

```
##.p8
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
*0008# 0004 0001 0004 0001 blk 0500 ab596000 ab9c7020 ab988bf0 led0 01 pmsHELL
##dd %ab988bf0 +74
%ab988c64 00000000 00070000 00041304 12d2ca34
%ab988c74 12d2ded8 00000000 00000000 00000000
%ab988c84 00000000 00000000 00000000 00000000
%ab988c94 00000000 00000000 00000000 00000000
%ab988ca4 00000000 00000000 00000000 00000000
%ab988cb4 00000000 00000000 00000000 00000000
%ab988cc4 00000000 00000000 00000000 00000000
%ab988cd4 00000000 00000000 00000000 00000000
```

The **TCB** address is found from the **.P** output.

Offset +0x74 into the **TCB** is the saved thread local memory area.

Offset +0x08 into the **TLMA** are the **AAB** registers.

The first is the last PM error to occur on this thread. In this case severity 4 error code **1304**.

The next double-word is the **PMQ**.

We can verify this by displaying it and checking the offset +0a4 is the same thread slot number.

### Notes:

After fix-pack 7 the TCB in WARP is extended by 4 bytes. The **TLMA** begins at **TCB+0x78**.

Since **AAB** is allocated using **DosAllocThreadLocalMemory** its location in the **TLMA** is depended how many **TLMA** allocations are made prior to PM initialising in the thread. In practice this is normally found at **TLMA** offset +0x8, but if the **AAB** is displaced it can usually be located by searching the **TLMA** for the **MQ** address which is normally in the range **12000000** to **14000000**.

```
##dd %12d2ca34
%12d2ca34 00000000 00000020 00000064 12d2cadc
%12d2ca44 12d2d75c 12d2cd3c 12d2cd3c 80002fff
%12d2ca54 80008000 00000004 00000001 00000001
%12d2ca64 80030038 0032bd01 000000ce 00000050
%12d2ca74 00000001 00000000 00000000 00000010
%12d2ca84 12d2c974 00000000 00000000 00000000
%12d2ca94 00000000 00000000 00000000 00000000
%12d2caa4 00000000 00000000 80000006 00000006
##d
%12d2cab4 00000000 00005453 0000024f 00000000
%12d2cac4 12d2c910 0bff0c02 00000000 00000000
%12d2cad4 00000001 00000008 80000007 00002f43
%12d2cae4 00000004 00000128 0000dc92 00010000
%12d2caf4 00000000 00000000 80000007 00002f43
%12d2cb04 00010001 00000128 0001168e 00010000
%12d2cb14 00000000 00000000 80000007 00002f43
%12d2cb24 00000004 00000128 0001168e 00010000
```

## How to Find the MQ of a BadApp Application

This example illustrates how to find the **MQ** of the application that causes the **BadApp** dialog to appear.

As discussed in [Application Not Responding to Messages Logic](#) **pmqsyslock**, **pmqfocus** and the **User\_Sem** PM semaphore owner will be reset when the **BadApp** dialog is displayed.

To find the **MQ** of the bad application under these circumstances we look at the Query Hung Process Structure (QHPSTRCUT).

```
##db fbadappdialog l1
9f3f:0000035c 01
##dd pmqsyslock l1
9f3f:0000ed14 00000000
##dd pmqfocus l1
9f3f:0000e0fc 12d2b0f0
##dd %12d2b0f0
%12d2b0f0 12d2b344 00000020 0000000a 12d2b198
%12d2b100 12d2b2d8 12d2b198 12d2b198 00002fff
%12d2b110 00010001 00000004 0000000f 00000001
%12d2b120 8003004a 0032e98f 00000021 00000157
%12d2b130 00000001 00000000 00000000 00000000
%12d2b140 00000000 00000000 00000000 00000000
%12d2b150 00000000 00000000 00000000 00000000
%12d2b160 00000000 00000000 80000021 00000000
##d
%12d2b170 10ff0000 00005453 0000024f 00000000
%12d2b180 12d2b08c 0bfff0002 00000000 00000000
%12d2b190 00000000 00000018 00000000 00000000
%12d2b1a0 00000000 00000000 00000000 00000000
%12d2b1b0 00000000 00000000 00000000 00000000
%12d2b1c0 00000000 00000000 00000000 00000000
%12d2b1d0 00000000 00000000 00000000 00000000
%12d2b1e0 00000000 00000000 00000000 00000000
##.p 18
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0018 0004 0001 0004 000f blk 0500 ab5b6000 ab9c7020 ab98ab70 1ed0 01 pmsshell
```

**fbadappdialog** is non-zero, which indicates that the **BadApp** dialog has been displayed.

**pmqsyslock** is not owned.

**pmqfocus** points to a shell thread, in fact the **BadApp** dialog thread.

So we look at **qhpsbadapp**

```
##dw qhpsbadapp l4
9f3f:0000e490 0002 000e 0008 0016

##.p
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0001 0001 0000 0000 0001 blk 0100 ffe38000 ffe3aa04 ffe3a80c 1eb4 00 *ager
0002 0001 0000 0000 0002 blk 0200 ab58a000 ffe3aa04 ab988020 1f3c 00 *tsd
0003 0001 0000 0000 0003 blk 0200 ab58c000 ffe3aa04 ab988218 1f50 00 *ctxh
0004 0001 0000 0000 0004 blk 081f ab58e000 ffe3aa04 ab988410 1f48 00 *kdb
0005 0001 0000 0000 0005 blk 0800 ab590000 ffe3aa04 ab988608 1f20 00 *lazyw
0006 0001 0000 0000 0006 blk 0800 ab592000 ffe3aa04 ab988800 1f3c 00 *asynchr
*0008# 0004 0001 0004 0001 blk 0500 ab596000 ab9c7020 ab988bf0 1ed0 01 pmsshell
000a 0004 0001 0004 0002 blk 0800 ab59a000 ab9c7020 ab988fe0 1ed4 01 pmsshell
000b 0004 0001 0004 0003 blk 0800 ab59c000 ab9c7020 ab9891d8 1ea8 01 pmsshell
000c 0004 0001 0004 0004 blk 0800 ab59e000 ab9c7020 ab9893d0 1ea8 01 pmsshell
000d 0004 0001 0004 0005 blk 0800 ab5a0000 ab9c7020 ab9895c8 1eb0 01 pmsshell
0010 0004 0001 0004 0006 blk 0200 ab5a6000 ab9c7020 ab989bb0 1edc 01 pmsshell
0011 0004 0001 0004 0007 blk 0200 ab5a8000 ab9c7020 ab989da8 1edc 01 pmsshell
0012 0004 0001 0004 0008 blk 0200 ab5aa000 ab9c7020 ab989fa0 1eb8 01 pmsshell
0007 0004 0001 0004 0009 blk 0200 ab594000 ab9c7020 ab9889f8 1ea8 01 pmsshell
0013 0004 0001 0004 000a blk 0800 ab5ac000 ab9c7020 ab98a198 1eb8 01 pmsshell
0014 0004 0001 0004 000b blk 0800 ab5ae000 ab9c7020 ab98a390 1eb8 01 pmsshell
0015 0004 0001 0004 000c blk 0800 ab5b0000 ab9c7020 ab98a588 1eb8 01 pmsshell
0016 0004 0001 0004 000d blk 0804 ab5b2000 ab9c7020 ab98a780 1ea8 01 pmsshell
0017 0004 0001 0004 000e blk 0804 ab5b4000 ab9c7020 ab98a978 1eb0 01 pmsshell
0018 0004 0001 0004 000f blk 0500 ab5b6000 ab9c7020 ab98ab70 1ea8 01 pmsshell
001a 0004 0001 0004 0010 blk 0200 ab5ba000 ab9c7020 ab98af60 1ed0 01 pmsshell
0009 0005 0004 0005 0001 blk 0800 ab598000 ab9c761c ab988de8 1eb4 00 harderr
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
000e 0005 0004 0005 0002 blk 0800 ab5a2000 ab9c761c ab9897c0 1ebc 00 harderr
000f 0005 0004 0005 0003 blk 0800 ab5a4000 ab9c761c ab9899b8 1eb8 00 harderr
```

```

0019 0007 0004 0007 0001 blk 0500 ab5b8000 ab9c8214 ab98ad68 led0 11 pmsshell
001c 0007 0004 0007 0002 blk 0800 ab5be000 ab9c8214 ab98b350 ledc 11 pmsshell
001d 0007 0004 0007 0003 blk 080a ab5c0000 ab9c8214 ab98b548 ledc 11 pmsshell
001e 0007 0004 0007 0004 blk 0800 ab5c2000 ab9c8214 ab98b740 led0 11 pmsshell
0020 0007 0004 0007 0005 blk 0800 ab5c6000 ab9c8214 ab98bb30 ledc 11 pmsshell
0021 0007 0004 0007 0006 blk 0200 ab5c8000 ab9c8214 ab98bd28 ledc 11 pmsshell
0025 0007 0004 0007 0007 blk 0200 ab5d0000 ab9c8214 ab98c508 led0 11 pmsshell
0026 0007 0004 0007 0008 blk 0200 ab5d2000 ab9c8214 ab98c700 ledc 11 pmsshell
0027 0007 0004 0007 0009 blk 0200 ab5d4000 ab9c8214 ab98c8f8 ledc 11 pmsshell
0029 0007 0004 0007 000b blk 0300 ab5d8000 ab9c8214 ab98cce8 led0 11 pmsshell
002a 0007 0004 0007 000c blk 021f ab5da000 ab9c8214 ab98cee0 leac 11 pmsshell
002b 0007 0004 0007 000d blk 0200 ab5dc000 ab9c8214 ab98d0d8 leb8 11 pmsshell
002f 0007 0004 0007 000e blk 0800 ab5e4000 ab9c8214 ab98d8b8 led0 11 pmsshell
001b 0006 0004 0006 0001 blk 0200 ab5bc000 ab9c7c18 ab98b158 1f00 10 pmspool
001f 0006 0004 0006 0002 blk 0200 ab5c4000 ab9c7c18 ab98b938 ledc 10 pmspool
0022 0006 0004 0006 0003 blk 0200 ab5ca000 ab9c7c18 ab98bf20 1e6c 10 pmspool
0023 0006 0004 0006 0004 blk 0200 ab5cc000 ab9c7c18 ab98c118 ledc 10 pmspool
0024 0006 0004 0006 0005 blk 0200 ab5ce000 ab9c7c18 ab98c310 ledc 10 pmspool
0028 0008 0004 0008 0001 blk 0200 ab5d6000 ab9c8810 ab98caf0 led0 12 cometrn
002c 0008 0004 0008 0002 blk 0801 ab5de000 ab9c8810 ab98d2d0 ledc 12 cometrn
002d 0009 0004 0009 0001 blk 0200 ab5e0000 ab9c8e0c ab98d4c8 led0 13 fpwmon
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
002e 000b 0004 000b 0001 blk 0400 ab5e2000 ab9c9408 ab98d6c0 leb8 15 cmd
0030 000e 0004 000e 0001 blk 0200 ab5e6000 ab9c9a04 ab98dab0 led0 16 turkey
0034 000e 0004 000e 0002 blk 0200 ab5ee000 ab9c9a04 ab98e290 led0 16 turkey
0038 000e 0004 000e 0003 blk 0200 ab5f6000 ab9c9a04 ab98ea70 led0 16 turkey
0037 000e 0004 000e 0004 blk 0200 ab5f4000 ab9c9a04 ab98e878 led0 16 turkey
0031 000e 0004 000e 0005 blk 0200 ab5e8000 ab9c9a04 ab98dca8 led0 16 turkey
0032 000e 0004 000e 0006 blk 0200 ab5ea000 ab9c9a04 ab98dea0 led0 16 turkey
0033 000e 0004 000e 0007 blk 0200 ab5ec000 ab9c9a04 ab98e098 led0 16 turkey
0035 000e 0004 000e 0008 crt 0500 ab5f0000 ab9c9a04 ab98e488 1f10 16 turkey
0036 000e 0004 000e 0009 blk 0500 ab5f2000 ab9c9a04 ab98e680 led0 16 turkey
0039 000e 0004 000e 000a blk 0200 ab5f8000 ab9c9a04 ab98ec68 led0 16 turkey
003a 000e 0004 000e 000b blk 0200 ab5fa000 ab9c9a04 ab98ee60 led0 16 turkey
003b 000e 0004 000e 000c blk 0200 ab5fc000 ab9c9a04 ab98f058 led0 16 turkey
003c 000e 0004 000e 000d blk 0200 ab5fe000 ab9c9a04 ab98f250 led0 16 turkey
003d 000e 0004 000e 000e blk 0200 ab600000 ab9c9a04 ab98f448 led0 16 turkey
003e 000e 0004 000e 000f blk 0200 ab602000 ab9c9a04 ab98f640 led0 16 turkey

```

The **QHPSTRUCT** shows Tid 2, Pid e, flags 6 and SGID 16

.P shows this to be slot 34.

If we use the technique described in [How to find the MQ of any thread](#) we will find the **MQ** for the bad application.

-----

## Finding Application and System Queue Elements

This example shows how to find the queue element on both the system queue and an application queue.

A similar technique applies to both types of queue. The system queue header is located from the address at **psysqueue**. Location of application queue headers has been discussed in [How to find the MQ of any thread](#).

The queue header contains the current read and write pointers, the queue element length and number of elements queued.

We illustrate this with the system queue in the following example:

```

##dd psysqueue l1
deff:00000000 1bdf0ac0
##dd 1bdf0ac0
1bdf0ac0 00000000 0030001e 00000078 1bdf0ae4
1bdf0ad0 1bdf18f4 1bdf1840 1bdf0fd0 00060000
1bdf0ae0 00070007 00000072 00510196 000002fe
1bdf0af0 00342420 1c0a9c00 01040040 00335362
1bdf0b00 00700040 015c0000 000000c1 26cf0000
1bdf0b10 1c000034 00401c0a 53c00104 00000033
1bdf0b20 00000071 00c1015c 000082fe 003426cf
1bdf0b30 1c0a9c00 01040040 003353ff 00700040
##dw 1bdf1840

```

```

%1bdf1840 0070 0000 0134 0050 0000 0000 1616 0034
%1bdf1850 8e00 0e7f 0040 0104 50f1 0033 0040 0071
%1bdf1860 0000 0134 0050 82fe 0000 172f 0034 1c00
%1bdf1870 1c0a 0040 0104 51cc 0033 0000 0072 0000
%1bdf1880 0134 0050 02fe 0000 180a 0034 9c00 1c0a
%1bdf1890 0040 0104 522a 0033 0040 0070 0000 019f
%1bdf18a0 0052 0000 0000 22c8 0034 1c00 1c0a 0040
%1bdf18b0 0104 5268 0033 0000 0071 0000 019f 0052
##

```

MQ+0x4 tells us 0x30 elements are queued, of length 0x1e bytes each.

MQ+0x14 is the current read pointer.

Displaying the queue from the current read pointer we can read off the first few message IDs since they are located at +0x0 of each entry: 70, 71, 72, and so on.

In an application queue the element length is 0x20.

-----

## PM Worked Examples Under OS/2 2.x

Dealing with PM application problems under OS/2 2.x is similar to WARP. The principle difference being that the messaging and windowing function in PM is provided by the 16-bit DLL, PMWIN.DLL.

Most of the message structures are analogous to those of PMMERGE.DLL, their layouts are similar.

Under PMWIN.DLL most pointers are either offsets from a predefined segments or selectors. Thus where there are double-word pointers in PMMERGE.DLL structures, there are word length fields in PMWIN.DLL.

The following three symbol files are required for debugging PM applications problems under OS/2 2.x:

- PMWIN.SYM
- PMGRE.SYM
- PMSHAPI.SYM

A selection of useful symbols in the OS/2 2.x PM environment, with their equivalent OS/2 3.0 is listed below:

OS/2 3.0	OS/2 2.x
pmqlist	<b>smqlist</b>
<b>pmqsyslock</b>	<b>smqsyslock</b>
<b>pmqfocus</b>	<b>smqfocus</b>
<b>pmqshell</b>	<b>smqshell</b>
<b>pmqshell2</b>	<b>smqshell2</b>
<b>pwndfocus</b>	<b>pwndfocus</b>
<b>fBadAppDialog</b>	<b>fBadAppDialog</b>
<b>qhpsBadApp</b>	<b>qhpsBadApp</b>

Another significant difference between the two environments is in the calling conventions:

- PMMERGE APIs use the 32-bit C calling convention
- PMWIN APIs use the 16-bit Pascal calling convention

In effect this means that parameters on stacks, and in some control blocks, are stored in reverse order.

There are four symbols that do not have equivalents in PMMERGE, these are:



winsel	The selector for the <b>AAB</b> regs segment for a process.
selsms	The selector for the <b>SMS</b> segment.
vphheapwnd	The table of <b>WND</b> heap pointers.
fsrsuser	The PM FastSafe RAMSEM, which is equivalent to the User_Sem PM Semaphore of PMMERGE.

We now run through a brief sequence of examples that illustrate:

[Finding an MQ and AAB registers](#)

[Finding an SMS from an MQ](#)

[Finding the WND from an HWND](#)

[Finding a BadApp process an MQ](#)

[Finding the System Queue](#)

## Finding an MQ and AAB Registers

```
##.s 8
##.p 8
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
*0008# 0006 0001 0006 0001 blk 0500 7b936000 7bb460d0 7bb28a58 1eb8 01 pmshell
##dw winsel 11
fd17:00000032 003f
##dd 3f:0
003f:00000000 00000000 00000000 00000000 00000000
003f:00000010 00081037 0000ebe7 ff3f0000 00000000
003f:00000020 00000000 00000000 00000000 00000000
003f:00000030 00000000 00000000 00000000 00000000
003f:00000040 00000000 00000000 00000000 00000000
003f:00000050 00000000 00000000 00000000 00000000
003f:00000060 00000000 00000000 00000000 00000000
003f:00000070 00000000 00000000 00000000 00000000
##d
003f:00000080 00000000 00000000 00000000 00000000
003f:00000090 00000000 0000e92f 00000000 00000000
003f:000000a0 00000000 0000f827 00000000 00000000
003f:000000b0 00000000 0000e957 00000000 00000000
003f:000000c0 00000000 0000e94f 00000000 00000000
003f:000000d0 00081001 0000f81f e8ff0000 00000000
003f:000000e0 00000000 00000000 00000000 00000000
003f:000000f0 00000000 0000e947 00000000 00000000
##dd ebe7:0
ebe7:00000000 001a0000 00640000 0aaa0082 06e806e8
ebe7:00000010 80002fff 80018001 00010006 06d60001
ebe7:00000020 bf5108b3 02252549 016a0000 00010000
ebe7:00000030 00000000 00000004 00000000 00000000
ebe7:00000040 00000000 00000000 00000000 00000000
ebe7:00000050 00000000 00000000 09d70000 000023cc
ebe7:00000060 00000010 54530000 0000022a 018c0000
ebe7:00000070 1802ec6f 00000bff 00010000 00080000
##dw ebe7:0
ebe7:00000000 0000 001a 0000 0064 0082 0aaa 06e8 06e8
ebe7:00000010 2fff 8000 8001 8001 0006 0001 0001 06d6
ebe7:00000020 08b3 bf51 2549 0225 0000 016a 0000 0001
ebe7:00000030 0000 0000 0004 0000 0000 0000 0000 0000
ebe7:00000040 0000 0000 0000 0000 0000 0000 0000 0000
ebe7:00000050 0000 0000 0000 0000 0000 09d7 23cc 0000
ebe7:00000060 0010 0000 0000 5453 022a 0000 0000 018c
ebe7:00000070 ec6f 1802 0bff 0000 0000 0001 0000 0008
```

**WinSel** gives the **AAB** segment selector.

**Note:**

**WinSel** is allocated in instance data, so must be viewed from a thread slot of the process in question.

Each entry is 0x10 bytes, one for each thread of the process.

The first entry is reserved.

The first double word of each entry is the past PM error for that thread and the second double word contains the selector for the **MQ** of that thread.

The key fields of interest in the **MQ** are:

Offset	Description
+0x0	chain pointer
+0x2	Queue element length
+0x4	number of elements queued
+0x6	Queue depth
+0x8	Top of queue
+0xa	Bottom of queue
+0xc	Current read pointer
+0xe	Current write pointer
+18	Pid
+1a	Tid
+1c	SGID
+30	SMS on which we are waiting a response
+32	SMS currently dispatched to our window procedure
+78	SMS at head of received list
+7e	thread slot id

## Finding an SMS From an MQ

```
##dw smgsyslock l1
fd9f:000003d4 e55f
##dw e55f:0
e55f:00000000 e567 001a 0000 000a 0082 0186 0082 0082
e55f:00000010 a400 0006 0006 0006 003d 0009 001e 072a
e55f:00000020 08b3 db25 2549 00f5 0000 005d 0000 0001
e55f:00000030 0094 0000 0010 0000 0000 0000 0000 0000
e55f:00000040 0000 0000 0000 0000 0000 0000 0000 0000
e55f:00000050 0000 0000 0000 0000 0000 0000 4d24 0000
e55f:00000060 0000 0000 ec37 5453 061c 0000 0000 6c4c
e55f:00000070 ec6f 0002 0bff 0000 0000 0001 0000 0048
##.s 48
##.p 48
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0048# 003d 0006 003d 0009 blk 0500 7b9b6000 7bb51cc4 7bb2f758 1eb8 1e turkey
##dw selsms l1
fd9f:00001c2a ec5f
```

```

##dw ec5f:94
ec5f:00000094 0000 0094 0000 0000 db25 2549 e55f e557
ec5f:000000a4 0000 0000 0020 0000 0000 0023 002b 0071
ec5f:000000b4 6d2c ec6f 0001 0024 0000 0000 3ae8 253a
ec5f:000000c4 e8df ebe7 0000 0000 0002 1b70 0410 0005
ec5f:000000d4 0000 0051 156c ec6f 0001 0038 01a8 0008
ec5f:000000e4 003c 003c 0082 0172 0000 0000 712c 1ec0
ec5f:000000f4 0172 0082 003c 003c 0172 0082 01ae 00be
ec5f:00000104 0000 0000 0000 0000 0002 0045 0003 0000

##dw e557:0
e557:00000000 e55f 001a 0000 000a 0082 0186 0082 0082
e557:00000010 2fff 0400 0400 0400 003d 000a 001e 072e
e557:00000020 08b3 98f9 2529 0000 0000 0000 0000 0000
e557:00000030 0000 0000 0010 0000 0000 0000 0000 0000
e557:00000040 0000 0000 0000 0000 0000 0000 0000 0000
e557:00000050 0000 0000 0000 0000 0000 0000 4dc0 0000
e557:00000060 0000 0000 ec37 5453 061c 0000 0000 6ce0
e557:00000070 ec6f 0002 0bff 0000 0094 0001 0000 0049

```

The thread with the system queue locked is waiting for a response to **WinSendMsg**. MQ+0x30 has the sent SMS offset.

The SMS selector is found from **selsms**.

The key fields in the **SMS** are:

Offset	Description
+0x0	Chain pointer offset
+0xc	Sending MQ selector
+0xe	Receiving MQ selector
+0x1a	Message Parameter 2
+0x1c	Message Parameter 1
+0x1e	Message Id
+0x20	Offset to WND
+0x22	Selector to WND

In this example the message has been sent to slot 49.

We see that the message has yet to be dispatched since it is still queued on the receive list (MQ+0x78).

-----

## Finding a WND From an Hwnd

```

##dw hwnddesktop l2
fd9f:0000053a 013c 0020

##dw vphheapwnd l2
fd9f:00001610 00ae ec6f

##dw ec6f:ae
ec6f:000000ae 0000 ec6f 0000 0000 0000 0000 0000 0000
ec6f:000000be 0000 0000 0000 0000 0000 0000 0000 0000
ec6f:000000ce 0000 0000 0000 0000 0000 0000 0000 0000
ec6f:000000de 0000 0000 0000 0000 0000 0000 0000 0000
ec6f:000000ee 0000 0000 0044 0000 0000 0000 0000 7098
ec6f:000000fe ec6f 0000 0000 0000 0000 0000 0000 0000
ec6f:0000010e 0000 0000 0000 1ffc 0000 ebe7 0010 165e
ec6f:0000011e fd2f 0065 1fad 0000 0000 0000 0000 0000

```

```

##dw ec6f:13c
ec6f:0000013c 0000 0000 0000 0000 1250 ec6f 0000 0000
ec6f:0000014c 0000 0000 0400 0300 0004 0000 a000 0000
ec6f:0000015c 1fd8 0000 ebe7 0020 3a24 fd4f 0065 1fad
ec6f:0000016c 0000 0000 0000 0000 0000 0000 0000 0000
ec6f:0000017c 0000 0000 0000 0000 0000 0000 0000 0048
ec6f:0000018c 0354 ec6f 00f4 ec6f 0000 0000 0000 0000
ec6f:0000019c 0000 0000 0000 0000 0000 0000 8000 0000
ec6f:000001ac 1e4c 0000 ebe7 0030 1d42 fd2f 0065 1fad

```

In this example we find the **WND** for the desktop from the **HWND** which is stored at **hwnddesktop**.

The **HWND** comprises an offset concatenated with an identifier, the low order nibble of which is a heap index. Thus, for the desktop:

```

##dw hwnddesktop 12
fd9f:0000053a 013c 0020
                . . .
                . . .
                . . .
offset...      . .
id.....      .
index.....

```

**vphheapwnd** points to a table of heaps. Each entry is a far pointer and there are at most 16. The index nibble of the **HWND** is used to select the heap pointer. In this example there is just one entry: **ec6f:0000**

We use the offset from the **HWND** with the heap selector to get the **PWND**. In this case **ec6f:13c**.

The key fields of interest in the **WND** are:

Offset	Description
+0x0	Next Sibling WND far pointer
+0x4	Parent WND far pointer
+0x8	Child WND far pointer
+0xc	Owner WND far pointer
+0x24	MQ selector that services this window
+0x26	ID and Index portion of the HWND for this WND.
+0x28	16-bit far pointer to the Window Procedure.
+0x2c	32-bit pointer to the Window Procedure.

-----

## Finding a BadApp Process and MQ

```

##db fbadappdialog 11
%1f8d07ae 01
##dw qhpsbadapp
%1f8d16b8 0002 003d 000a 001e 000c 0000 0000 0000
%1f8d16c8 0000 0000 0000 0000 0000 0000 0000 0000
%1f8d16d8 0000 0000 0000 0000 0000 0000 0000 0000
%1f8d16e8 0000 0000 0000 0000 0000 0000 0000 0000
%1f8d16f8 0000 0000 0000 0000 0000 0000 0000 0000
%1f8d1708 0000 0000 0000 0000 0000 0000 0000 0000
%1f8d1718 0000 0000 0000 0000 0000 0000 0000 0000
%1f8d1728 0000 0000 0000 0000 0000 0000 0000 0000

```

The fields of the **QHPSTRUCT** are in a different order to the PMMERGE version:

Offset	Description
+0x0	Flags
+0x2	Pid
+0x4	Tid
+0x6	SGID

-----

## Finding the System Queue

```
##dw mghsysqueue
fd87:00000000 0000 001c 001a 0078 0ad6 17f6 0ee2 11ba
fd87:00000010 0000 0006 0007 0007 0000 0001 0001 ebe7
fd87:00000020 06d6 0008 0ba5 0000 0000 e55f 072a 0048
fd87:00000030 000f 000b 0002 0000 0300 0000 0000 0000
fd87:00000040 0000 fffe ee6f 2549 0000 0000 0000 0588
fd87:00000050 0339 032c 00ac 00bc 00cc 00dc 00ec 00fc
fd87:00000060 010c 012c 011c 013c 014c 015c 0000 0000
fd87:00000070 0000 0000 0000 0000 0000 0000 0000 0000
##dw ee2
fd87:00000ee2 0072 00f5 005d 02fe 0000 dbc2 2549 8000
fd87:00000ef2 0000 8000 0000 9150 08a8 08a8 0070 0032
fd87:00000f02 00ec 0000 0000 dfaa 2549 0000 c3b7 fff6
fd87:00000f12 2002 0000 0012 08a8 0071 0032 00ec 83fe
fd87:00000f22 0000 e007 2549 fff6 2002 0000 0012 08a8
fd87:00000f32 e007 08a8 0072 0032 00ec 03fe 0000 e065
fd87:00000f42 2549 2006 0000 0016 0000 2006 08a8 08a8
fd87:00000f52 0071 0032 00ec 82fe 0000 e0c3 2549 0000
```

**mghsysqueue** points directly at the system queue.

The queue pointers are offsets from the same segment as the MQ.

The current read pointer at +0x0e, is 0x0ee2.

The queue entry length at +0x02, is 0x001c.

Displaying the queue from the read pointer shows the first few elements queued are for message IDs, 72, 70, 71, 72 and so on.

Each entry on the system queue is a **SQMSG**. The key fields are:

Offset	Description
+0x0	Message ID
+0x2	Message Parameter 1
+0x6	Message Parameter 2

For an application queue, the entries are **QMSG** structures. the key fields of these are:

Offset	Description
+0x0	HWND
+0x4	Message Id
+0x8	Message Parameter 1
+0xc	Message Parameter 2

---

## Dump Analysis of Loops in Ring 0 Code

Ring zero loops can sometimes be successfully analysed from a dump. The trick is knowing how to locate the register set at the time the dump was taken.

The Dump Formatter only implements the .R command, which obtains the registers from a stack frame on the thread's ring 0 stack. Under the kernel Debugger there is no problem: the R command will display the current system registers.

### Note:

If a thread never runs in User Mode, such as the internal PID 0 threads then a stack frame is never built and .R will be unsuccessful in formatting the registers.

Fortunately there is a way of obtaining the current registers:

When a dump is initiated using Ctrl-Alt-Numlock-Numlock a keyboard interrupt is initiated by the processor hardware.

Via the IDT control passes to the interrupt router who is responsible for switching to the interrupt stack before passing control to the appropriate interrupt handler.

The interrupt router checks to see if the system is already running from the interrupt stack.

If it isn't then an interrupt stack frame is built on the current stack and the stack frame pointer is saved in **fpoldstack**. Then the SS selector is switched to the interrupt stack selector (E8).

If it is then a nested interrupt has occurred and the interrupt stack frame is built on the interrupt stack itself.

It is from **fpoldstack** that we are able to obtain the registers before any interrupt occurred. The following debug log illustrates this and many of the techniques previously discussed.

---

## Ring 0 Loop Dump Analysis Example

This example finds a loop in a file system driver from a system dump. For reference, we note the format of the interrupt stack frame as pointed to by **fpoldstack** as follows:

+0x0	Current interrupt level when prior to interrupt.
+0x4	GS
+0x8	FS
+0xc	ES
+0x10	DS
+0x14	EDI
+0x18	ESI
+0x1c	EBP
+0x20	padesp
+0x24	EBX
+0x28	EDX
+0x2c	ECX

```

+0x30      EAX
+0x34      pad
+0x3c      EIP
+0x40      CS
+0x44      EFLAG
+0x48      ESP
+0x4c      SS
+0x4c      SS

```

>>Who's the current thread?

```

# .p*
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
*00a3# 006c 000a 006c 0001 run 0200 7b720000 7bb025c0 7ba8f9e0 0894 25 FRNOLBMG

```

>>Probably a loop of some kind, could be a hot I/O or even dispatcher bug (unlikely).

>> Where are we?

```

# .r
eax=001fe624 ebx=00002022 ecx=00000029 edx=00000007 esi=00000000 edi=0003e77c
eip=00000179 esp=0003e624 ebp=0003e68c iopl=2 -- -- -- nv up ei pl nz na po nc
cs=d02f ss=001f ds=0053 es=0053 fs=150b gs=0000 cr2=00000000 cr3=001bb000
d02f:00000179 66ea4102021a5b00 jmp      005b:1a020241
# .m

*har      par      cpg      va      flg next prev link hash hob      hal
00b1 %fee13f40 00000010 %1a050000 3d9 00b0 00b2 0000 0000 00bd 0000 hco=02c2c
hob      har hobnxt flgs own hmte sown,cnt lt st xf
00bd 00b1 0000 0838 00bb 00bb 0000 00 00 00 00 shared c:doscalle1.dll
hco=2c2c pco=ffe71cf7 hconext=02c20 hptda=18c9 f=1c pid=00bc c:cmd.exe

```

>> We are in DOSCALL1. Let's see what function was called.

```

# dw ss:bp
001f:0000e68c e6bc 0003 ae60 1a02 e77c 0003 e728 0003
001f:0000e69c e70c 0003 0000 0000 0000 0000 0010 0000
001f:0000e6ac 2022 0000 0000 0000 28a4 111b e697 0003
001f:0000e6bc e740 0003 4a6d 1113 e77c 0003 e728 0003
001f:0000e6cc e70c 0003 0000 0000 0000 0000 0010 0000
001f:0000e6dc 2022 0000 0000 0000 0010 0000 ab18 1111
001f:0000e6ec e77c 0003 0002 0000 0000 0000 0000 0000
001f:0000e6fc 0001 0000 0007 0000 0000 0000 0180 0000

```

```

# ln %11134a6d
No Symbols Found

```

```

# .m %11134a6d

```

```

*har      par      cpg      va      flg next prev link hash hob      hal
0e82 %fee26f36 00000030 %11110000 3d9 09af 09f1 0000 0000 15b1 0000 hco=02c17
hob      har hobnxt flgs own hmte sown,cnt lt st xf
15b1 0e82 0000 0838 08fe 08fe 0000 00 00 00 00 shared c:frnococl.dll
hco=2c17 pco=ffe71c8e hconext=02de0 hptda=1938 f=1c pid=00bb c:frnosa.exe

```

```

# .lmo 8fe
hmte=08fe pmte=%fdef0c68 mflags=4498b186 c:\frnvlr0\dl1\frnococl.dll
obj  vsize  vbase  flags  ipagemap cpagemap hob sel
0001 00026ee8 11110000 80002025 00000001 00000027 15b1 888f r-x shr big
0002 0000001f 11180000 80001025 00000028 00000001 0caf 88c7 r-x shr alias
0003 00000030 11190000 80002025 00000029 00000001 15a4 88cf r-x shr big
0004 00000008 111a0000 80001003 0000002a 00000000 0000 88d7 rw- alias
0005 00004e74 111b0000 80002003 0000002a 00000005 0000 88df rw- big

```

```

# u %11134a6d-10%11134a5d 085657      or      byte ptr [esi+57],dl
%11134a60 6a00      push      +00

```

```

%11134a62 51          push    ecx
%11134a63 8b4dac       mov     ecx,dword ptr [ebp-54]
%11134a66 52          push    edx
%11134a67 51          push    ecx
%11134a68 e8af63ef08   call    %1a02aelc
%11134a6d 8bc8       mov     ecx,eax
%11134a6f 8b45ac       mov     eax,dword ptr [ebp-54]
%11134a72 83c420       add     esp,+20      ; ' '
%11134a75 894ddc       mov     dword ptr [ebp-24],ecx
%11134a78 83f904       cmp     ecx,+04

# ln %1a02aelc
%1a02aelc DOSCALL1 DOS32OPEN

>> So we're in DOSOPEN

>> We've almost certainly call-gated into the kernel.
>> Check this out ...

# u cs:eip -10
d02f:00000169 268805       mov     byte ptr es:[di],al
d02f:0000016c 8bc3       mov     ax,bx
d02f:0000016e 8cc2       mov     dx,es
d02f:00000170 5f         pop     di
d02f:00000171 c9         leave
d02f:00000172 c3         ret
d02f:00000173 90         nop
d02f:00000174 9a00000a1b call    1b0a:0000
d02f:00000179 66ea4102021a5b00 jmp     005b:1a020241
d02f:00000181 9a00001a1b call    1b1a:0000
d02f:00000186 66ea3b05021a5b00 jmp     005b:1a02053b
d02f:0000018e 9a0000631b call    1b63:0000
# dg 1b0a
1b0a CallG32 Sel:Off=0148:0000529f      DPL=3 P   DWC=8
# ln 148:529f
0148:0000529f OS2KRNL DOSOPEN2
#

>> Yes, that's where we are.
>> Now let's see if we can find out where in R0 DOSOPEN has got...

# dw interruptlevel 11
0400:00006382 0000
# dd currintlevel 11
0148:0000529f OS2KRNL DOSOPEN2
#
>> So, no nested interrupts, but we are handling one
>> (interruptlevel=0000).
>> The current interrupt came from IRQ 1 (currintlevel=1)
>> So a keyboard interrupt, not surprising because the customer was
>> asked to take a dump using ctrl-alt-numlock-numlock, furthermore
>> he obeyed!

>> Lets look at the interrupt stack saved by the interrupt router
>> (prior to switching stacks).

# dd fpoldstack 12
%fff27310 00004b0c 00000030

>> So, the Interrupt Stack Frame maps the frame at 30:4b0c

# dw 30:4b0c
0030:00004b0c ffff ffff 0000 0000 03b8 0000 2410 0000
0030:00004b1c 23f8 0000 4c66 7b61 0000 7b61 4b5e 0003
0030:00004b2c 4b40 0000 0000 0000 24d0 0000 ffff 0000
0030:00004b3c 0000 0000 0000 0000 0000 0000 be93 0000
0030:00004b4c 23d0 0000 2292 0000 23f8 0000 006c 0000
0030:00004b5c 00a3 4b8c b541 23d0 23f8 0000 4c66 0000
0030:00004b6c 4baa 0029 2b00 0000 0004 0001 2101 2d76
0030:00004b7c 0001 0094 0022 003f 0000 0000 20d4 4be8

>> bp is at +1c, sp at +20, cs at +40, eip at +3c
>> We took the dump at when cs:eip=23d0:be93
>> who is this?

# .m 23d0:0

*har      par      cpg      va      flg next prev link hash hob      hal
0021 %feel32e0 00001400 %fc953000 121 0020 0022 0000 0020 0022 0000      =0000

```



```

hob   har hobnxt flgs own  hmte   sown,cnt lt st xf
0022  0021 0000  0225 ffe0 0000  0000 00  04 00 00 vmkshrw

```

```

>> 23d0 is allocated out of the Kernel Swappable Read/Write heap.
>> Lets see who owns this heap block. Need to look at the VMKSHB
>> shared heap block header...

```

```

# dg 23d0
23d0 Code Bas=fca15000 Lim=0000ff5f DPL=0 P RE A
# dw %face15000-10
%face15000 0000 0000 0000 0000 ff68 5200 ff4d 23d0
%face15000
Invalid linear address: %face15000

```

```

>> VMKSHB is an 8 byte prefix of the form:
>> ulong size || 0x520000
>> ushort hob
>> ushort sel

```

```

>> check the owner hob

```

```

# .mo ff4d
ff4d fsd6
#

```

```

>> This was allocated by/for the 6th loaded FSD.
>> N.B fsd8 is used for the 8th and subsequent FSDs
>> in the same way dd16 is used for the 16th and subsequent
>> device driver. From Fix Pack 35 of Warp 3 and GA Warp 4
>> the dd1-dd16 and fsd-fsd8 system object ids are not used, but
>> instead the hmte for the driver is used. This avoids the problem of
>> the non-uniqueness of dd16 and fsd8. It also directly identifies
>> the driver module.

```

```

>> Who is fsd6?
>> There are two ways to home in on this.

```

```

>> First method.
>> List all library modules (includes DLLs IFSs FONs etc)

```

```

# .lml
hmte=18e2 pmte=%felal000 mflags=0498b188 c:\frnvlr0\dll\frnolgar.dll
hmte=193e pmte=%felal3ac mflags=0498b188 c:\frnvlr0\dll\frnosars.dll
hmte=18fd pmte=%felal8d4 mflags=4498b186 c:\frnvlr0\dll\frnofo2.dll
hmte=1933 pmte=%felal94c mflags=4498b186 c:\frnvlr0\dll\frnofios.dll
hmte=18f2 pmte=%fdebd1c4 mflags=4498b186 c:\frnvlr0\dll\frnoutil.dll
hmte=0fc7 pmte=%fdedf550 mflags=4498b1c6 c:\frnvlr0\dll\frnollmn.dll
.
.

```

```

>> 200 hundred lines later

```

```

.
hmte=0b0a pmte=%fe0bb6f8 mflags=0408b1c8 c:\cmlib\dll\acshpres.dll
hmte=0af9 pmte=%fe0bb864 mflags=0408b1c8 c:\cmlib\redj.pml
hmte=0af6 pmte=%fe0bb8f0 mflags=0408b1c8 c:\cmlib\redj.pdl
hmte=08f7 pmte=%fe0bcf8c mflags=0408b1c8 c:\cmlib\redj2.pml
hmte=094d pmte=%felldc44 mflags=0408b1c8 c:\cmlib\redj2.pdl
hmte=0a8a pmte=%fe098ea4 mflags=0408b1c8 c:\cmlib\cm20sys.pml
hmte=0a87 pmte=%fe098f5c mflags=0408b1c8 c:\cmlib\cm20sys.pdl
hmte=04c7 pmte=%fde5ff60 mflags=0498b1c8 c:\os2\dll\times.fon
hmte=04c5 pmte=%fde5fa7c mflags=0498b1c8 c:\os2\dll\helv.fon
hmte=04c3 pmte=%fde5fb44 mflags=0498b1c8 c:\os2\dll\courier.fon
hmte=04c1 pmte=%fde5fbb4 mflags=0498b1c8 c:\os2\dll\sysmono.fon
hmte=04b9 pmte=%fde5fc8c mflags=4498b1c5 c:\os2\dll\pmatm.dll
hmte=0324 pmte=%fel134f54 mflags=0428alc9 d:\ibm3995\demoifs.ifs
hmte=02ff pmte=%fe0fff90 mflags=0428alc9 c:\netw\nwifs.ifs
hmte=01b9 pmte=%fe0dbcb4 mflags=0428alc9 c:\showcase\sdcfs.ifs
hmte=0109 pmte=%fe0befa0 mflags=0428alc9 c:\os2\cdfs.ifs
hmte=00e0 pmte=%fde46d3c mflags=0428alc9 c:\os2\hpfs.ifs
hmte=0076 pmte=%fde14f68 mflags=0428alc9 a:\mini_fsd.fsd

```

```

>> FSDs were installed in order, 76, e0, 109, 1b9, 2ff and 324.

```

```

>> fsd6 is therefore hmte=324. Lets check for certain:

```

```

# .lmo 324
hmte=0324 pmte=%fel134f54 mflags=0428alc9 d:\ibm3995\demoifs.ifs
seg sect psiz vsiz hob sel flags
0001 0003 d7ba d7ba 0000 2398 8d60 code shr prel rel

```

```

0002 0070 2325 2325 0000 23a0 8d60 code shr prel rel
0003 0083 ec66 ec66 0000 23a8 8d60 code shr prel rel
0004 00fa f7a6 f7a6 0000 23b0 8d60 code shr prel rel
0005 0176 ble4 ble4 0000 23b8 8d60 code shr prel rel
0006 01d0 f3e6 f3e6 0000 23c0 8d60 code shr prel rel
0007 024b eeda eeda 0000 23c8 8d60 code shr prel rel
0008 02c3 ff60 ff60 0000 23d0 8d60 code shr prel rel
0009 0343 fe82 fe82 0000 23d8 8d60 code shr prel rel
000a 03c3 4b4e 4b4e 0000 23e0 8d60 code shr prel rel
000b 03e9 002a 002a 0000 23e8 8c60 code shr prel
000c 03ea 4298 4298 0000 23f0 8d60 code shr prel rel
000d 040c 9dfd 9dfe 0000 23f8 8d41 data prel rel
000e 0000 0000 1964 0000 2400 8c41 data prel
000f 0000 0000 fe88 0000 2408 8c41 data prel
0010 0000 0000 d75e 0000 2410 8c41 data prel

```

>> and yes we find selector 23d0 in object 8 of demoifs.ifs

>> Second method

>> We could approach this from the FSC control block, which is similar  
>> in purpose to the DD header chain.  
>> The FSC is a table of FSD entry point tables. We might spot the  
>> selector in question being referenced in the FSC. If not, we can  
>> unwind the R0 stack until we do find a reference.

>> First the FSC. Dump the SAS for the FSC selector

```

# .a
--- SAS Base Section ---
        SAS signature: SAS
        offset to tables section: 0016
        FLAT selector for kernel data: 0158
        offset to configuration section: 001E
        offset to device driver section: 0020
        offset to Virtual Memory section: 002C
        offset to Tasking section: 005C
        offset to RAS section: 006E
        offset to File System section: 0074
        offset to infoseg section: 0080
--- SAS Protected Modes Tables Section ---
        selector for GDT: 0008
        selector for LDT: 0000
        selector for IDT: 0018
        selector for GDTPPOOL: 0100
--- SAS Device Driver Section ---
        offset for the first bimodal dd: 0CB9
        offset for the first real mode dd: 0000
        sel for Drive Parameter Block: 0520
        sel for ABIOs prot. mode CDA: 0468
        seg for ABIOs real mode CDA: 6800
        selector for FSC: 00C8
--- SAS Task Section ---
        selector for current PTDA: 0030
        FLAT offset for process tree head: FFF29714
        FLAT address for TCB address array: FFF26BDA
        offset for current TCB number: FFE23A0E
        offset for ThreadCount: FFE23A12
--- SAS File System Section ---
        handle to MFT PTree: FDE55FB4
        selector for System File Table: 00C0
        sel. for Volume Parameter Bloc: 0678
        sel. for Current Directory Struc: 06A8
        selector for buffer segment: 00A8
--- SAS Information Segment Section ---
        selector for global info seg: 0428
        address of curtask local infoseg: 03B80000
        address of DOS task's infoseg: FFFFFFFF
        selector for Codepage Data: 06BB
--- SAS RAS Section ---
        selector for System Trace Data Area: 0508
        segment for System Trace Data Area: 0508
        offset for trace event mask: 09D6
--- SAS Configuration Section ---
        offset for Device Config. Table: 0D40
--- SAS Virtual Memory Mgt. Section ---
        Flat offset of arena records: FFF2C314
        Flat offset of object records: FFF2C32C
        Flat offset of context records: FFF2C31C
        Flat offset of kernel mte records: FFF27E68

```

Flat offset of linked mte list: FFF273B8  
Flat offset of page frame table: FFF2A768  
Flat offset of page range table: FFF29CC0  
Flat offset of swap frame array: FFF260B0  
Flat offset of Idle Head: FFF294D4  
Flat offset of Free Head: FFF294C4  
Flat offset of Heap Array: FFF2A770  
Flat offset of all mte records: FFF2BE24

#

>> FSC selector is c8. Now dump the FCS segment.

# dw c8:0

00c8:00000000 03c8 0000 0000 fde1 0b68 0738 0b6c 0738  
00c8:00000010 0000 0720 01fc 0720 0010 0720 05b4 0718  
00c8:00000020 0570 0720 0580 0720 0634 0720 0640 0720  
00c8:00000030 0e3c 0720 1120 0720 0834 0720 090c 0720  
00c8:00000040 09f8 0718 1130 0720 1f24 0720 1f6e 0720  
00c8:00000050 2122 0720 16e4 0720 1b10 0720 1b38 0720  
00c8:00000060 1bec 0720 1dc8 0720 0c60 0718 0d70 0718  
00c8:00000070 1f14 0720 215c 0720 22a0 0720 2294 0720

# d

00c8:00000080 111c 0718 25fc 0720 26b0 0720 117c 0718  
00c8:00000090 0fdc 0718 0000 0718 0000 0000 0000 0000  
00c8:000000a0 0000 0000 03b0 0720 062c 0718 137c 0720  
00c8:000000b0 0bb4 0718 26bc 0720 0000 0000 0000 0000  
00c8:000000c0 0000 0000 0000 0000 0000 0a58 0004 0a58  
00c8:000000d0 0000 0a50 01b6 0a50 000e 0a50 04c4 0a50  
00c8:000000e0 060e 0a50 061c 0a50 065c 0a50 0666 0a50  
00c8:000000f0 1198 0a50 1224 0a50 086e 0a50 0e0e 0a50

# d

00c8:00000100 09e6 0a50 125a 0a50 299a 0a50 29a8 0a50  
00c8:00000110 29b6 0a50 263e 0a50 278c 0a50 27aa 0a50  
00c8:00000120 2842 0a50 288a 0a50 28fc 0a50 298c 0a50  
00c8:00000130 297e 0a50 29c4 0a50 2cf6 0a50 2cb8 0a50  
00c8:00000140 2f0c 0a50 34b2 0a50 34f2 0a50 350c 0a50  
00c8:00000150 5029 1000 9cab 0140 1232 0a50 1240 0a50  
00c8:00000160 0000 0000 0602 0a50 9d8c 0140 1568 0a50  
00c8:00000170 124e 0a50 3500 0a50 0000 0000 0000 0000

# d

00c8:00000180 0000 0000 0000 0000 0090 1028 008a 1028  
00c8:00000190 00be 1018 03e6 1018 05da 1018 0766 1018  
00c8:000001a0 09ee 1018 0c38 1018 0db6 1018 0dc2 1018  
00c8:000001b0 0fa4 1018 1488 1018 1496 1018 158a 1018  
00c8:000001c0 1998 1018 1cae 1018 1f92 1018 1fa0 1018  
00c8:000001d0 1fae 1018 2998 1018 2bf0 1018 2c9c 1018  
00c8:000001e0 2caa 1018 2f90 1018 2f9e 1018 31c4 1018  
00c8:000001f0 332c 1018 333a 1018 3950 1018 3e9c 1018

# d

00c8:00000200 3fa2 1018 419a 1018 4318 1018 4332 1018  
00c8:00000210 5029 1000 9cab 0140 0000 0000 0000 0000  
00c8:00000220 0000 0000 08aa 1018 9d8c 0140 2114 1018  
00c8:00000230 1fbc 1018 4326 1018 0000 0000 0000 0000  
00c8:00000240 0000 0000 0000 0000 0098 22a8 0090 22a8  
00c8:00000250 01b0 22b0 1500 22b0 7470 22b0 1650 22b0  
00c8:00000260 18e0 22b0 51b0 22b0 25d0 22b0 344d 22b0  
00c8:00000270 3ca7 22b0 469a 22b0 28ac 22b0 279b 22b0

# d

00c8:00000280 4316 22b0 28c5 22b0 4343 22b0 4347 22b0  
00c8:00000290 434d 22b0 3b30 22b0 43c0 22b0 4353 22b0  
00c8:000002a0 195d 22b0 4310 22b0 6e50 22b0 4c10 22b0  
00c8:000002b0 4d20 22b0 4ec6 22b0 5bc2 22b0 4359 22b0  
00c8:000002c0 6d27 22b0 1aab 22b0 43a1 22b0 8740 22b0  
00c8:000002d0 5029 1000 9cab 0140 4a70 22b0 4a7f 22b0  
00c8:000002e0 84c0 22b0 17bb 22b0 9d8c 0140 37f0 22b0  
00c8:000002f0 25f0 22b0 7570 22b0 0000 0000 0000 0000

# d

00c8:00000300 0000 0000 0000 0000 0c96 23f8 8880 23f8  
00c8:00000310 00de 23a0 0152 23a0 01bd 23a0 0228 23a0  
00c8:00000320 02f2 23a0 0369 23a0 03d1 23a0 042d 23a0  
00c8:00000330 049e 23a0 050f 23a0 0580 23a0 05df 23a0  
00c8:00000340 066b 23a0 06eb 23a0 075f 23a0 07b5 23a0  
00c8:00000350 083e 23a0 0982 23a0 09e7 23a0 0a4c 23a0  
00c8:00000360 0acf 23a0 0b40 23a0 0bab 23a0 0c1f 23a0  
00c8:00000370 0c87 23a0 0cfb 23a0 0d8d 23a0 0e04 23a0

# d

00c8:00000380 0e5d 23a0 0ecb 23a0 0f33 23a0 0f92 23a0  
00c8:00000390 5029 1000 9cab 0140 0000 0000 0000 0000  
00c8:000003a0 0000 0000 028d 23a0 9d8c 0140 0905 23a0

```

00c8:000003b0 08ac 23a0 1000 23a0 0000 0000 0000 0000
00c8:000003c0 0000 0000 0000 0000
Past end of segment: 00c8:000003c8
#
>> The FSD starts with an 8 byte header. Word 1 is the length.
>> Each entry is for each FSD starting with fsd2 (fsd1 is OS2BOOT
>> and not used once the kernel is loaded). Each FSD entry comprises
>> a table of far16 pointers. The first two are a) pointer to FSD
>> attributes and b) FSD name. The remaining are the function entry
>> points (See IFS OEM reference). There are 46 of these. In other
>> words the first fsd entry is at c8:8 and ever subsequent entry is
>> every 12 lines of display. fsd 6 entry starts at c3:308

>> what's fsd6 called?

# db 23f8:8880 18
23f8:00008880 4f 50 54 4c 49 42 00 00 OPTLIB..

>> The evidently is the optical library fsd.
>> We didn't find the current cs:eip in the fsd function table so
>> we unwind the r0 stack ....

# dw 30:4b8c 18
0030:00004b8c 4be8 7426 23a8 4bb2 0030 4bb0 0030 4bd8
# dw 30:4be8 18
0030:00004be8 4c0a 738a 23a8 0000 1004 0000 2f48 4c2c
# dw 30:4c0a 18
0030:00004c0a 4c3a bb28 23a8 d7ce 0001 4c2c 0030 0000
# dw 30:4c3a 18
0030:00004c3a 4c6c 1b8f 23b0 0000 0000 0000 d7ce 0001
# dw 30:4c6c 18
0030:00004c6c 4c7e 1e87 23b0 0300 24f8 4ca8 0030 23f8
# dw 30:4c7e 18
0030:00004c7e 4cca 08ed 23b0 0300 24f8 4ca8 0030 4cb0
# dw 30:4cca 18
0030:00004cca 4cf6 011e 23b0 0100 24f0 0073 2520 0000
# dw 30:4cf6 18
0030:00004cf6 4d36 7314 23b0 04d3 2408 0073 2520 0000
# dw 30:4d36 18
0030:00004d36 4d70 5be8 23b0 006a 2520 0000 04d3 2408
# dw 30:4d70 18
0030:00004d70 4dbc 04e3 23a8 0000 04d3 2408 0020 2022
# dw 30:4dbc 18
0030:00004dbc 4e46 2b1f 2398 4eb6 0030 0000 0003 0020
# dw 30:4e4d 18
0030:00004e4d 304e 0000 0300 2000 bc00 304e 1000 2200
# dw 30:304e 18
Invalid linear address: 0030:0000304e

>> The problem here is that the kernel is not using ebp
>> before calling the fsd. So dump the R0 stack from
>> the last recognisable fsd selector. Look for the
>> first selector that matches one used in fsd6's
>> function table.

dw 30:4dbc
0030:00004dbc 4e46 2b1f 2398 4eb6 0030 0000 0003 0020
0030:00004dcc 4ebc 0030 0010 2022 0000 41a5 2cf0 4df2
0030:00004ddc 0030 ffff 04d0 2408 4edb 0030 0000 2f40
0030:00004dec 23f8 8bfc 0308 2022 0000 1004 8ce8 1c63
0030:00004dfc 8ce8 1c63 8ce8 1c63 0000 0000 0000 0000
0030:00004e0c 0000 006c 0000 0274 0000 0000 0000 2100
0030:00004e1c 0001 4e3e 0006 0004 0000 2f40 039c 2408
0030:00004e2c 0345 0098 006c 0003 0001 0000 04d0 2408
# d
0030:00004e3c 02f4 0305 0098 2f40 1004 4e84 0d6e 23a0
0030:00004e4c 4eb6 0030 0000 0003 0020 4ebc 0030 0010
0030:00004e5c 2022 0000 41a5 2cf0 41d7 2cf0 ffff 4fee
0030:00004e6c 0030 4edb 0030 4ee3 0030 0308 8bfc 510d
0030:00004e7c 0000 9410 00c8 0030 4ec0 9a09 0140 4eb6
0030:00004e8c 0030 0000 0003 0020 4ebc 0030 0010 2022
0030:00004e9c 0000 41a5 2cf0 41d7 2cf0 ffff 4fee 0030
0030:00004eac 4edb 0030 4ee3 0030 0000 0000 04d0 0007

>> at 30:4e48 we have 23a0:d6e. Looking at the function
>> table we see entry point 26 at 23a0:cfb is the closest.
>> fsd entry point 26 is FS_OPENCREATE. This seems to be
>> consistent with what ring 3 was doing.

```

>> Finally for future reference the FSD entry structure is  
>> as follows:

```
>>+0  FS_ATTRIBUTE; /* -> FSD attribute. (in FSD memory) */
>>+4  FS_NAME;      /* -> FSD name. (in FSD memory) */
>>+8  FS_ATTACH;    /* DosQFsAttach, DosFsAttach */
>>+c  FS_CHDIR;     /* DosChdir */
>>+10  FS_CHGFILEPTR; /* DosChgFilePtr */
>>+14  FS_CLOSE;    /* DosClose */
>>+18  FS_COPY;     /* DosCopy */
>>+1c  FS_DELETE;   /* DosDelete */
>>+20  FS_EXIT;     /* DosExit */
>>+24  FS_FILEATTRIBUTE; /* DosFileInfo, DosSetFileMode */
>>+28  FS_FILEINFO;  /* DosQFileInfo, DosSetFileInfo */
>>+2c  FS_FILEIO;    /* DosFileIO */
>>+30  FS_FINDCLOSE; /* DosFindClose */
>>+34  FS_FINDFIRST; /* DosFindFirst */
>>+38  FS_FINDFROMNAME; /* DosFindFromName-Private to server */
>>+3c  FS_FINDNEXT;  /* DosFindNext */
>>+40  FS_FINDNOTIFYCLOSE; /* DosFindNotifyClose */
>>+44  FS_FINDNOTIFYFIRST; /* DosFindNotifyFirst */
>>+48  FS_FINDNOTIFYNEXT; /* DosFindNotifyNext */
>>+4c  FS_FSINFO;    /* DosQFsInfo, DosSetFsInfo */
>>+50  FS_INIT;     /* -- No corresponding API */
>>+54  FS_IOCTL;    /* DosDevIoctl */
>>+58  FS_MKDIR;    /* DosMkdir */
>>+5c  FS_MOUNT;    /* -- No corresponding API */
>>+60  FS_MOVE;     /* DosMove */
>>+64  FS_NEWSIZE;  /* DosNewsize */
>>+68  FS_NMPIPE;   /* All named pipe related API's */
>>+6c  FS_OPENCREATE; /* DosOpen */
>>+70  FS_PATHINFO; /* DosQPathInfo, DosSetPathInfo */
>>+74  FS_PROCESSNAME; /* -- No corresponding API */
>>+78  FS_READ;     /* DosRead, DosReadAsync */
>>+7c  FS_RMDIR;    /* DosRmdir */
>>+80  FS_SETSWAP;  /* -- No corresponding API */
>>+84  FS_WRITE;    /* DosWrite, DosWriteAsync */
>>+88  FS_OPENPAGEFILE; /* init time only */
>>+8c  FS_ALLOCATEPAGESPACE; /* size swap file */
>>+90  FS_CANCELLOCKREQUEST; /* DosCancelLockRequest */
>>+94  FS_FILELOCKS; /* DosSetFileLocks */
>>+98  FS_VERIFYUNCNAME; /* Used to save function addresses */
>>+9c  FS_COMMIT;   /* DosBufReset, DosClose */
>>+a0  FS_DOPAGEIO; /* perform paging */
>>+a4  FS_FSCTL;    /* DosFsCtl */
>>+a8  FS_FLUSHBUF; /* DosBufReset */
>>+ac  FS_SHUTDOWN; /* DosShutdown */
>>+b0  FS_SDCHGFILEPTR; /* Used to save function addresses */
>>+b4  FS_SDFSINFO;  /* at shutdown time. These functions */
>>+b8  FS_SDREAD;    /* will only be called by shutdown */
>>+bc  FS_SDWRITE;   /* filters. */
>>
>> * Bit masks for FS_ATTRIBUTE (remember FS_ATTRIBUTE points to the
>>attribute
>> * word rather than containing it directly.)
>>
>> FS_ATTR_REMOTE 0x0001 /* 0 = local FSD, 1 = remote FSD */
>> FS_ATTR_UNC 0x0002 /* 0 = normal, 1 = this is UNC FSD */
>> FS_ATTR_LOCKINFO 0x0004 /* 0 = no notice, 1=notify filelocks */
>> FS_ATTR_LVL7 0x0008 /* 0 = no level 7 requests, 1 = yes */
>> FS_ATTR_PIPESVR 0x0010 /* 0 = don't FSD on PIPE req, 1 = yes */
>>
>> /* bit masks for FS_ATTRIBUTE (High Word) */
>> FS_ATTR_VERN0 0x7000 /* bits 28-30 version no */
>> FS_ATTR_EA 0x8000 /* bit 31 -> 1 = extended attribute */
>>
>> /* equates for commit type */
>> FS_COMMIT_ALL 2 /* all handles commit */
>> FS_COMMIT_ONE 1 /* one handle commit */
>>
>> /* equates for close type */
>> FS_CL_ORDINARY 0 /* ordinary close */
>> FS_CL_FORPROC 1 /* final close for process */
>> FS_CL_FORSYS 2 /* final close for system */
-----
```

---

# Kernel Debugger User Guide

The Kernel Debugger is essentially a replacement OS/2 Kernel module that contains an in-built [debugger](#) component. With the debugger one may halt system execution, inspect and alter memory and registers and display system control blocks. The debugger is controlled from a dump ASCII terminal (the debugging console) which is connected to the machine under test (MUT), either directly or via a modem-modem link, through one of its COMx ports. The debugger supports a comprehensive command set, which is fully described in the [Kernel Debugger and Dump Formatter Command Reference](#).

The debug kernel is distributed in two forms:

## ALLSTRICT

This version of the kernel contains all optional self diagnostic (otherwise known as strict or assertion checking) code. Besides this functional difference many of the system control blocks have extra accounting and signature fields. This has a number of consequences that may affect problem diagnosis:

- 1 Performance characteristics will be different since extra checking and accounting is being performed.
- 2 Memory usage will be different because of extra diagnostic code, extensions to system control blocks and in some cases additional space to cause page faults rather than overlays by erroneous code.
- 3 Timing critical problems might not be re-createable under the **ALLSTRICT** kernel.
- 4 Secondary problems may be detected or even introduced through the use of additional diagnostic code.

## HSTRICT

This version of the kernel is essential the **RETAIL** kernel with the debugger component. It contains only a limited set of strict checking code. The system control blocks are of the same form as those used by in the **RETAIL** kernel. The performance characteristics of the **HSTRICT** kernel are closer to those of the **RETAIL** kernel than the **ALLSTRICT** kernel. For this reason the **HSTRICT** kernel is recommended as a first choice when diagnosing application and non-system problems.

The base version of the **ALLSTRICT** kernel is distributed with the **OS/2 Developer's Toolkit**. Versions of the **HSTRICT** and **ALLSTRICT** kernels for fix packs may be obtained from the following sources:

- The OS/2 Base Product CDROM for WARP is distributed with the **ALLSTRICT** kernel and Dump Formatter. (For the initial release of WARP this was only available on the US version of WARP).
- The Developer Connection CDROM - this may be ordered through the Developer Assistance Program (DAP) or the System Library Subscription Service (SLSS).
- From your local IBM Marketing Representative.
- From the World Wide Web at URL:

<ftp://service.boulder.ibm.com>

- For IBM customers from by FTP from the node:

<ftp.software.ibm.com>, directory `ps\products\os2\fixes\debug`

- For IBM internal users by FTP to the SDM at node:

<sdm.austin.ibm.com>

Logon using Id and Password **Anonymous**. A list of files is contained in **files.bbs**.

## Note:

The Kernel Debugger packages obtainable from the SDM are equipped with an installation procedure and two text files **dbsetup.txt** and **modemset.txt** that give instructions on how to install the debugger for local and remote debugging via a modem.

---

## Kernel Debugger Local Set-up

The following items are required to install and set up a local debug session:

Either the **HSTRICT** or **ALLSTRICT** kernel appropriate to the level of the MUT.

System symbol files. These are optional, but useful [breakpoints](#) and system data are difficult to locate without them.

Application symbol files. These are only necessary if you intend to debug complex applications where data and subroutines are difficult to locate without them.

System Trace definition and Formatting files. These are only required if you intend to trace [kernel dynamic tracepoints](#) while using the debug kernel.

A null modem cable.

An asynchronous ASCII dumb terminal or an emulator on another PC. Softterm, which is distributed with OS/2 is suitable. [PMDf](#), which is part of the **OS2PDP** package distributed with this book also provides a terminal emulator interface suitable for use with the Kernel Debugger. Other popular emulators used with the Kernel Debugger include: **PMDEBUG**, **DEBUGO** and **LOGICOMM**.

Confusion sometimes arises over the installation of the kernel debugger, particularly as the OS/2 Developer's Toolkit distributes debug versions of other OS/2 modules. Note in particular:

The debug versions of **OS2LDR**, **PMDD.SYS**, **PMGRE.DLL** and **PMWIN.DLL** are optional. These modules will route additional diagnostic information to the debug console if they are installed.

No modification of **CONFIG.SYS** is required.

A secondary console attached to the MUT may not be used as a debug console.

---

## Installing the Debug Kernel

If you use the OS/2 Developer's toolkit to install the debug kernel then the installation is performed automatically using the supplied **DBGINST** command. If you choose to install the debug kernel manually then perform the following steps:

1. Copy the debug kernel (OS2KRNL or OS2KRNLB) to the root directory of the boot drive.
2. Copy the symbol files into the same directories as their corresponding load modules. Usually system symbol files are distributed on a diskettes that have the same directory structure as OS/2 system code. This conveniently allows the [UNPACK](#) command to be used to copy all symbols files in one operation (per diskette).
3. Unhide the **RETAIL** kernel module using the following command:  

```
ATTRIB -r -s -h OS2KRNL
```
4. Rename the **RETAIL** kernel to something unique, e.g **OS2KRNLr**
5. Rename the **ALLSTRICT** or **HSTRICT** kernel to **OS2KRNL**. There is no need to hide or make the replaced kernel read-only, unless you wish to protect yourself against accidents!

The MUT is now ready to use in debug mode as soon as it is re-booted. Before that happens the debug console needs to be set up.

### Note:

It is possible to run the MUT with the debug kernel installed without setting up the debug console. This is particularly useful when diagnosing pervasive problems. If the COM port settings are correct when the problem reoccurs then the debug console may be connected at that time.

-----

# Debug Terminal Set-up

This section describes the connection and set-up of the debugging console. You may need to know the operational requirements of both you local COM port (on the MUT) and dumb ASCII terminal. Fortunately the debug kernel does not impose any form of hand-shaking or a fixed COM speed setting. In many cases default settings apply. First we discuss the cable requirements.

A null modem cabel is required to connect the MUT to the debug console. This is essentially a 3-wire circuit that connects the two COM connectors together. Some PCs are equipped with a 25-pin sockets, other 9-pin. A null modem cable is a symetric circuit so we do not distinguish which is the MUT and which the console.

## 25-to-25 Pin Cable

MUT / CONSOLE DB25J	CONSOLE / MUT DB25J
2	3
3	2
7	7

## 25-to-9 Pin Cable

MUT / CONSOLE DB25J	CONSOLE / MUT DB9J
2	2
3	3
7	5

## 9-to-9 Pin Cable

MUT / CONSOLE DB9J	CONSOLE / MUT DB9J
2	3
3	2
5	5

### Note:

The three connections involved are:

- RX (receive)
- TX (transmit)
- SG (Signal Ground)



The null modem cable essentially connects RX-TX and SG-SG. The pin conventions for RX and TX on a 25-pin connector reverse those of a 9-pin connect. Thus the 25-9 connection looks like a non-null circuit.

If you intend to debug on a number of different set-ups then it is worth equipping yourself with the following items, which are commercially available:

- Standard modem cable.
- A gender changer.
- A null modem convertor.
- A 25-9 pin convertor.

With these items you should be able to cater for most variations and remote connection as well.

The next thing to consider is the COM port settings. By default the debug kernel will first select COM2. If that is in use then COM1. If you require the debugger to use another COM port, or a non-standard I/O port address then you might need to set this explicitly by using the **.B command**, which should be entered in the **KDB.INI** initialisation file.

By default the kernel debugger initialises the selected COM port to run at 9600 bits per second. If your debugging console requires a different speed setting then you should convey this to the debug kernel using the **.B command**, again entered in the **KDB.INI** file.

The default communications protocol uses 8 data bits, 1 stop bit and no parity. If this needs to be different then it may be set using the **O command** also entered in the **KDB.INI** file.

Finally some COM ports require the **DTR** signal to be held high before allowing communication. If this is necessary then it can be set using the debug kernel to write to the I/O port that controls the COM port set-up register. This may be done using the **O command** entered in the **KDB.INI** file.

Examples of using these commands in **KDB.INI** is given in the next section.

Having set up the COM port requirements on the MUT the debug console must be set up to match. Precisely how this is done will depend on whether a dumb terminal or terminal emulator software is used. If you use emulator software under OS/2 you may need to use the **OS/2 MODE command** to select compatible COM port settings for the debugging console's COM port.

---

## The KDB.INI Initialisation File

The debug kernel normally only accepts commands entered at the debugging console. However, during system initialisation it will accept commands entered into a text file, which if used, must be called **KDB.INI** and reside in the root directory of the boot drive.

The **KDB.INI** file is read after the kernel has loaded and the kernel symbols are loaded and the system is running in protect mode.

### Warning:

The content of the **KDB.INI** file is somewhat sensitive. If you make a syntax or format error then you may hang the system and have to re-boot from installation diskettes to recover.

On most systems the use of a KDB.INI file is not required to establish correct operation of the COM port and should be avoided.

Each command must be terminated with a <CR><LF> pair **except the last in the file**.

The **KDB.INI** is most easily created using:

```
COPY CON: KDB.INI
```

Enter the commands you require, using the <RETURN> key after each command except the last. For the final command, terminate it using the sequence: Ctrl-Z <RETURN>.

### Note:

Use of an editor for creating **KDB.INI** may not be suitable if the <CR><LF> sequence cannot be suppressed from the last line.

The following example shows how to select COM3 at 1200 bps, with DTR held high and to prepare the debugger to intercept any ring 2 or 3 traps.

```
.b 1200t 3e8
O 3ec 1
vsf *
g
```

#### Notes:

Since the default arithmetic base for the debugger is hexadecimal a **t** suffix is required if the COM port speed is specified in decimal as in the example.

We have assumed a standard port address assignment for COM3, namely 3e8 for data register and 3ec for control register.

The [VSF command](#) causes the debugger to intercept all ring2 and ring3 traps and give control to the debug console.

The [G command](#) is required unless you want to enter the debugger as soon as the kernel has entered protect mode, loaded its symbol file and executed the **KDB.INI** file.

---

## Kernel Debugger Remote Set-up

This section describes how to use the kernel debugger remotely, that is with a modem-modem link between the machine under test (MUT) and the debugging console.

The first step is to install the debug kernel and symbols files on the MUT as described preceding section, [Kernel Debugger Local Set-up](#)

Although the Debug Kernel will work with nearly any modem, configuration details are unique to each modem. This topic describes the setup of several modems, and gives general guidelines for setting up others.

---

## Items Required to Setup a System for Remote Debugging

To complete the installation, you will need:

- The RETAIL and either the HSTRICT or ALLSTRICT Kernel
- A modem
- A modem data cable
- An analog dial-in telephone line
- Communications software.

---

## Modem

Most asynchronous modems currently available will be suitable for use as a remote-debug modem. For best performance, the modem should:

- Support auto-answer operation
- Support locked DTE speed at 9600 bps

- Allow connections at CCITT V.32 (9600 bps), and V.22bis (2400 bps)
- Support error-correction (MNP or V.42)
- Save configuration so a power-outage does not lose settings.

-----

## Modem data cable

The configuration of the cable used to connect the modem to the MUT is not important. Any serial data cable should have the connections required by the debug kernel. Just make sure you don't use a **null-modem** cable. You will either need a 25-to-25 pin cable (for connection to the built-in serial port on a PS/2), or a 25-to-9 pin cable (for connection to a 9-pin serial port).

Required connections for remote debug cable:

### 25-to-25 Pin Cable

MODEM DB25P	COMPUTER DB25J
2	2
3	3
7	7

### 25-to-9 Pin Cable

MODEM DB25P	COMPUTER DB9J
2	3
3	2
7	5

Notice the 25-to-9 pin cable reverses pins 2 and 3. Do not confuse this with a null-modem cable - the signals on a 25-to-9 pin cable are normally reversed.

-----

## Analog Dial-in Telephone Line

In order to call the modem and connect to the MUT, you will need a standard voice-grade telephone line that can be direct-dialled. A connection can be made if the line must go through a switchboard, but it makes it more difficult for the person doing the debugging. Digital telephone lines won't work at all with the modem.

-----

## Communications Software

Any terminal software that can communicate at 9600 bps will do. OS/2 2.0 comes with a program (Softterm Custom) that is adequate. **PMDf**, which is part of the **OS2PDP** package on the CDROM that accompanies this book, also provides a terminal emulation facility but in addition

provides **REXX** support that allows Kernel Debugger command sequences to be automated.

-----

## The Configuration Process

After you have assembled the required items, follow these steps to prepare the MUT for remote debugging:

1. Connect the Modem to the MUT.

Connect one end of the data cable to the modem, and the other end to the serial port on the MUT. If the MUT has more than one serial port, connect the cable to the port configured as COM2 (the debug kernel uses COM2 by default). On PS/2 systems, the reference diskette can tell you which port is configured as COM2. Connect the telephone line to the modem, and power the modem on.

2. Program the modem for DEBUG operation:

Programming the modem may be a complex process, depending on the type of modem and the intended use. There are two ways to program the modem:

- Quick programming for single debug use
- Full programming for "permanent" debug use.

The "quick" method is simple, but the modem will not be programmed to recover from loss of power or repeated calls. The "full" method allows the modem to be programmed once, and then used whenever debugging is needed.

The "quick" programming is performed by the debug kernel itself through use of the **KDB.INI** file. In addition to containing start up commands for the debugger **KDB.INI** can also contain modem initialisation strings coded as operands to the Kernel Debugger [? command](#). For this reason, the modem must be connected and powered on when the MUT is booted, and cannot be powered off until debugging is complete.

The first lines of the **KDB.INI** may will be COM port selection and parameters if defaults are not suitable, for example:

**.B 1200t 1**

(Set debugger for 1200 bps, comm port 1)

Following this are the modem initialisation strings, which are unique to each type of modem. The commands in the initialisation string must:

- Activate "auto-answer"
- Lock the DTE at 9600 bps
- Activate XON/XOFF flow control
- Ignore the DTR signal (not supplied by the debug kernel)
- Suppress result codes.

The remaining lines of the **KDB.INI** file may contain other debugging commands. The last of these is normally **G**.

The "quick" programming strings for several popular modems are as follows.

**? "AT&F E0 Q1 &B1 &H2 &I2 &D0 S0=1"**

US Robotics HST and Dual Standard

**? "AT&F2 E0 Q2 &D0 &K4 S0=1"**

Supra FAX/Modem V.32bis

**? "AT&F E0 Q1 &D0 \Q1 S0=1"**

Intel 14.4EX

An alternative "quick" technique for entering the Hayes modem initialisation commands, which avoids the use of **KDB.INI** is illustrated by the following example. This example assumes that the default COM2 port is to be used:

1. In **CONFIG.SYS** add the following line

```
RUN=C:\OS2\CMD.EXE /K C:\MODEM.CMD
```

2. Edit a file called MODEM.CMD and enter the following two lines

```
MODE COM2:9600,N,8,1
COPY MODEM COM2
```

3. Edit a file called MODEM and enter the following line

```
AT&K4&D0S0=1&W
```

To use "Full" programming, you will configure the modem with the same features as in "quick" programming, but the settings will be stored in the modem's firmware (or set in modem switches). Determining how to store these settings can be difficult. A thorough study of the modem manual may be required. To program the modem, use a terminal emulation program (for example, the SOFTERM program that is supplied with OS/2). When programming the modem, Set the terminal program for 9600 BPS operation, and type the appropriate modem string. Since the initialisation string instructs the modem to suppress result codes, the modem will not return a response. The "FULL" programming strings for several modems are:

**AT&F &B1 &H2 &I2 &W**

US Robotics HST and Dual Standard

**AT&F2 E0 Q2 &D0 &K4 S0=1 &W**

Supra FAX/Modem V.32bis

**AT&F E0 Q1 &D0 \Q1 S0=1 &W**

Intel 14.4EX

NOTE: The US Robotics HST Dual Standard does not store all settings, but has external switches instead. After programming the modem, set the switches as follows:

**1=ON**

(DTR forced ON)

**2=don't care**

(result code type)

**3=OFF**

(result code suppressed)

**4=ON**

(command echo suppressed)

**5=OFF**

(auto-answer enabled)

**6=don't care**

(carrier detect function)

**7=ON**

(result code in originate mode only)

**8=ON**

(AT commands enabled)

**9=ON**

(don't disconnect for +++)

**10=OFF**

(load NVRAM at power-on)

**QUAD=OFF**

(normal connect - ON if null modem cable used)

Once the modem is connected, and programmed, the system should be ready for remote debugging. Re-boot the system with the debug kernel installed. When the telephone rings, the debug modem should answer the phone, and establish connection with the caller. The modem-to-kernel speed should remain at 9600 bps (the default speed used by the debug kernel), but the modem-to-modem speed can be whatever is used by the remote modem. If both modems support error correction, correction will be used.

-----

## Using Low-speed Modems

If a 9600 bps modem is not available, a slower modem can be used with the debug kernel. If the modem supports "speed conversion" (a 2400 bps modem with error-correction and compression will support speed conversion), setup is straightforward. Construct the proper initialisation string for the modem, making sure that the modem's DTE speed (modem-to-debugger) speed is locked at 9600 bps. If the modem does not support "speed conversion," construct an initialisation string for the modem, and create a **KDB.INI** file that resets the debugger to the speed supported by the modem. For example, **.B 2400t 2** for a 2400 bps modem. In this case, the person calling the debugger will have to use the speed supported by the modem.

## Limitations of This Setup

Since the modem communicates with the MUT at 9600 bps, but can communicate with the remote modem at any speed, the modem must use flow control to avoid data overruns. The only flow control supported by the debug kernel is [XON/XOFF](#). The only problem this causes is when the remote user wants to pause a continuous data display by pressing CTRL-S. If the modem has also sent a CTRL-S, the one from the user will be ignored. You may have to press CTRL-S several times before the display pauses. This is not a problem if the remote user's communications program supports a "scroll-back buffer," in which case there is no reason to pause the display with CTRL-S.

## Troubleshooting

If, after following these directions, you cannot establish a remote debug connection, this guide may help:

Symptom	Problem	Solution
Modem rings, but doesn't answer	Modem not set for auto-answer	check modem programming (look for AA light on modem).
... " ... " ...	Phone line not connected to modem	Plug in telephone line to modem.
Modem answers, but no response from debug kernel	Retail kernel installed	Remove RETAIL kernel and install DEBUG kernel
... " ... " ...	Data cable not connected properly	Connect data cable from modem to MUT. plug into COM2 if MUT has more than one serial port
User at the remote modem sees "garbage" on screen, unable to control debug session	Modem not locked at 9600 bps	check modem configuration
... " ... " ...	Debug Kernel not operating at 9600 BPS	Add <b>.B 9600T</b> to KDB.INI file (create file if needed, in root directory of boot drive). Re-boot MUT.

## Controlling the System From the Debugging Console

Having set up the Kernel Debugger for a [Local](#) or [Remote](#) debug session the system is ready to be controlled from the debugging console. The console is used in two modes, which for convenience we refer to as:

Monitor mode, and

Command mode

In Monitor mode the console acts merely as a output device for displaying diagnostic messages from the debug kernel and debug versions any other of system modules that write messages to the debugger's COM port. In this mode it is not possible to enter [Kernel Debugger commands](#) without having first switched to command mode. In monitor mode the system runs more or less as a retail system except for the performance overheads imparted by the additional diagnostic code.

Monitor mode is in effect initially unless a [KDB.INI](#) file is defined.

The console switches to monitor mode after [G command](#) is executed.

In Command mode normal system execution is suspended. The debug component of the kernel monitors the debugging console for command input and indicates this with using one of the following command prompts:

- > Signifies that the system has been suspended while in real mode.
- # Signifies that the system has been suspended while in protect mode with paging disabled.
- Signifies that the system has been suspended while in V86 mode with paging disabled.
- ## Signifies that the system has been suspended while in protect mode with paging enabled.
- Signifies that the system has been suspended while in V86 mode with paging enabled.

In addition to these prompts the Kernel Debugger also uses a data prompt when a commands require additional input. This is signified by a single colon prompt `:`. Commands such as [R](#) and [E](#) may use a data prompt.

Command mode is entered when one of the following events occur:

A fatal exception while executing in ring 0

Any unrecoverable exception occurring in a device driver, file system driver or the OS/2 kernel will result in a fatal error if it is allowed to be intercepted by the system exception handlers. When this occurs it is usually not possible to restore the system to a running state.

The [VTF command](#) may be used to intercept potentially fatal exceptions before the system's exception handlers receive control. If the exception condition is corrected manually then the system may continue to run after the [G command](#) is entered. See [Trap and Exception Processing](#) for further information.

An Internal Processing Error (IPE) occurs.

Internal processing errors are unrecoverable conditions that are detected by the OS/2 kernel. Some of these are exceptions (described in the previous bullet); others are inconsistencies that arise from invalid logical conditions or invalid system data. Under the retail kernel IPEs result in the system halting. Under the debug kernel the console enters command mode after an error message is displayed. IPE messages may be suppressed from displaying as a hard error popups by setting the byte at symbol: **fDebugOnly** to a non-zero value. Under the debug kernel some IPEs are generated for recoverable conditions and allow the system to continue execution after the [G command](#) is entered. An example of a recoverable IPE is where the loader detects a *bad* or mismatched symbol file for a module it is loading. When this occurs the system displays message:

```
Internal Symbol Error
```

Command mode is entered. If the [G](#) command is subsequently issued the system will be allowed to continue execution without the *bad* symbol file being activated.

A *sticky* [breakpoint](#) fires

*Sticky* breakpoints are set using the [BP](#) and [BR](#) commands. The system is may be returned to a running state after the [G command](#) is entered.

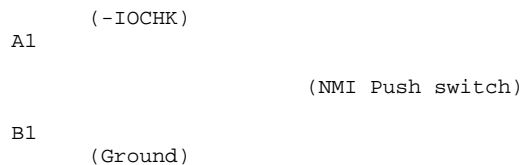
An unhandled non-maskable interrupt (NMI).

NMIs normally signal hardware error conditions. Under the RETAIL kernel these usually result in TRAP 2 fatal exceptions unless an NMI handler has been registered by a device driver. Under the debug kernel, unhandled NMIs cause control to be given to the debugging console from which it is possible to return the system to a running state using the [G command](#).

NMIs are may be generated from several sources, which include:

Channel Check	This occurs when an I/O card activates the channel check signal.
Memory parity error	This occurs when memory capable of parity bit generation, detects a parity discrepancy as memory is fetched from RAM.
DMA bus time-out.	This occurs when a DMA-driven device uses the bus for longer than the maximum allowed period of 7.8 microseconds.
The watchdog timer interrupt.	This occurs when the NMI watchdog (NWD) is enabled and timer interrupts (IRQ 0) are disabled causing loss of timer ticks. OS/2 maintains an NWD count, which if exceeds a maximum value then an IPE is generated. Some hardware/BIOS also maintains an NWD counter, but the precise details of the NWD mechanism are machine specific. For some systems the NWD may not be supported. For further information refer to the appropriate hardware and BIOS reference literature for the machine type under consideration.

Unless the NMI is masked off using by setting mask bit 0x80 in I/O port 0x70, the NMI channel check provides a means of breaking into the system even when it is disabled for (maskable) interrupts, that is, when the **CLI** instruction has been used to clear the interrupt flag in the **EFLAGS** register. An an ISA-bus system a prototype card may be used to implement the following circuit, which provides an NMI push-button switch:



**Note:**

OS/2 normally only disables NMIs during system initialisation and when the Kernel Debugger is running in command mode. However, the Kernel Debugger will allow only one attempt to break in using a channel check NMI, after which NMIs are disabled until the system is re-booted.

An **INT 3** instruction is executed

**INT 3** instructions are used by the system to implement tracing (see [The System Trace Facility](#)) and software [breakpoints](#). However any program may use **INT 3** instructions freely under the Kernel Debugger to cause system execution to be suspended and the debugging console to switch to command mode.

**Note:**

Under the **RETAIL** kernel, **INT 3** instructions other than those implemented by the system for tracing cause code to be terminated with a TRAP 3 exception.

The user enters **Ctrl-C** from the debugging console.

Unless the system is in a disabled state, the user may type **Ctrl-C** from the debugging console at any time to cause immediate suspension of normal system execution and the console to switch to command mode.

The user holds down the **r-key** from the debugging console at system initialisation time.

If the **r-key** is held down at system initialisation time the debugging console will switch to command mode shortly after the OS2KRNL has entered real-mode for the first time. At this time no symbols have been loaded, paging has never been enabled and the **KDB.INI** file has not been processed.

**Note:**

In real-mode many of the Kernel Debugger external commands are not available (because they rely on Virtual Memory Management to be initialised). **Attempts to use them may cause unpredictable results or even total system failure.**

The holds down the **p-key** at the debugging console at system initialisation time.

If the **p-key** is held down at system initialisation time the debugging console will switch to command mode shortly after the OS2KRNL has entered protect-mode for the first time. At this time no symbols have been loaded, paging is disabled and the **KDB.INI** file has not been processed.

The user holds down the **Space-bar** from the debugging console at system initialisation time.



If the **space-bar** is held down at system initialisation time the debugging console will switch to command mode shortly after the OS2KRNL has entered protect-mode and fully initialised. At this time OS2KRNL symbols have been loaded and paging is enabled but the **KDB.INI** file has not been processed.

The **KDB.INI** file is processed.

If the **KDB.INI** file is present then the Kernel Debugger effectively enters command mode by executing Kernel Debugger commands from the **KDB.INI** file. After the last command is executed, the command prompt appears at the debugging console, unless that last command was a **G** command.

---

## Controlling Output to the Debugging Console

In both monitor and command mode the following control key sequences are supported:

### Ctrl-C

Will cancel the currently running command and return the console to command mode.

### Ctrl-S

Will temporarily suspend output to the debugging console and suspend system execution.

### Ctrl-Q

Will resume system execution and output to the debugging console.

**Note:** **Ctrl-Q** and **Ctrl-S** correspond to the ASCII asynchronous communications control characters: **XON** and **XOFF**. These may be used by any terminal emulator, which interfaces with the the Kernel Debugger, as a data pacing mechanism.

---

## Optional System Diagnostic Facilities

Several system components implement optional diagnostic facilities under the debug kernel. These cause additional checking and in some cases detailed information to be displayed at the debugging console when certain debug flags switches are set.

### Note:

Debugging switches are not a formally architected feature of the OS/2 operating system. They are provided primarily for use by OS/2 developers in debugging and testing the system. They are therefore subject to change or withdrawal without any notice whatsoever.

In this section the following logging facilities are described:

[Forcing a System Dump from the Kernel Debugger](#)

[Virtual Memory Management Lock Trace](#)

[Virtual Memory Heap Validation](#)

[Loader Logging Facility](#)

[DosDebug Logging Facility](#)

[DosPTrace Logging Facility](#)

---

## Forcing a System Dump from the Kernel Debugger

Sometimes the situation arises where neither a kernel debug session or a system dump alone are sufficient to analyse a problem. Typically

this occurs with problems where evidence of the cause has been removed from the system before the problem occurrence becomes recognised but the problem itself requires lengthy analysis even when the causal conditions are intercepted. Examples of this are problems where:

Storage overlays, may not be noticed until the valid owner of the storage traps at some later time.

A program terminates apparently normally, but unexpectedly.

A deadlock or hang occurs because a resource owner *forgets* to release ownership of a shared resource.

If the problem is such there are readily identifiable criteria that allow it to be intercepted closer to its cause, for example by using [breakpoints](#) under the Kernel Debugger, then being able to take a dump at such a point can be advantageous.

The simplest technique for initiating a system dump is to enter the **.SYSDUMP command**, which is new from fix pack 29 for Warp 3.0 and base Warp 4.0. Prior to these releases other techniques have to be employed. The simplest of these is to type the dump key sequence (**Ctrl-Alt-Numlock-Numlock** or **Ctrl\_Alt\_F10\_F10**) from the keyboard of the system under test while the debugger is in console mode. Then type the **G command** from the debug console. The keyboard interrupt will be serviced and the standalone dump procedure initiated.

In an unattended situation a manually initiated dump may not be feasible. The following techniques discuss how to initiate the system dump in a more automated fashion. In some cases it may be possible to set up the command automation from the [KDB.INI](#) initialisation file.

The system dump is initiated when the kernel routine **RASRST** (RAS restart) is called. Normally this occurs from ring 0 when exception management intercepts a trap and **TRAPDUMP** is coded in the **CONFIG.SYS** file or when the keyboard device driver (**KDB.SYS**) intercepts a **Ctrl-Alt-Numlock-Numlock** or **Ctrl-Alt-F10-F10** sequence. From ring 3 **RASRST** is called indirectly via the **Dos32ForceSystemDump** API since **RASRST** is not addressable from any user code selectors. The Kernel Debugger **G command** allows an address to be specified where execution is to continue from, which provides a means calling the system dump routine from the debugging console. Before using this technique, the following points must be understood:

**RASRST** is not addressable from user code selectors since they have an upper address boundary of at most 512M.

**RASRST** requires to be executed using a 16-bit code selector.

**RASRST** requires a ring 0 stack selector to be active

**Dos32ForceSystemDump** requires a 32-bit code selector, such as **5b**.

On some early versions of OS/2 2.1 **Dos32ForceSystemDump** is unreliable.

The symbol **Dos32ForceSystemDump** occurs in both DOSCALL1.DLL and the callgate entry point in OS2KRNL.

From ring 0 the following command will generally be successful in initiating a system dump:

```
g =rasrst
```

From ring 2 or ring 3, 32-bit code the following commands will be successful providing **Dos32ForceSystemDump** is working correctly. The address of DOSCALL1:DOS32FORCESYSTEMDUMP is determined first, then a call to **Dos32ForceSystemDump** is made:

```
ln dos32forcesystemdump
%1a027c78 doscall1:FLAT32:DOS32FORCESYSTEMDUMP

g =1a027c78
```

For 16-bit application code the **CS** register must be set to a value that will address DOSCALL1.DOS32FORCESYSTEMDUMP. A suitable selector would be **5b** for ring-3 code and **5a** for ring-2. So, for 16-bit code this procedure becomes:

```
ln dos32forcesystemdump
%1a027c78 doscall1:FLAT32:DOS32FORCESYSTEMDUMP
r cs 5b (or r cs 5a)

g =1a027c78
```

If **TRAPDUMP** is in effect then a dump can be forced by causing an immediate trap. The most effective way to achieve this is to set the current **SS** selector to 0 using the [R command](#). For example:

```
r ss=00
g
```

If you wish to trap an application the very next time it runs in user mode then use **.R** to determine the user registers and set a **breakpoint** on **CS:EIP** in the context of the application's **thread slot** and specify that **SS** be set to zero when the breakpoint fires. For example:

```
.p 2d
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
002d  000b  0002  000b  0001  blk  0200  7b700000  7b8c68fc  7b8acb60  1eb8  14  mrfilepm

##.r 2d
eax=00000000 ebx=00000000 ecx=0000aa37 edx=0000a9ef esi=00090bff edi=00090000
eip=0000272d esp=0000b228 ebp=0009b230 iopl=2 -- -- -- nv up ei ng nz na pe nc
cs=d02f ss=004f ds=a9ef es=be47 fs=150b gs=0000 cr2=1704b000 cr3=001d9000
doscall1:CONFORM16:postDOSSEMWAIT:
002d|d02f:0000272d c9          leave          ;br0

##bp d02f:272d,"j wo(tasknumber)==2d, '.r:r ss=0;g'ig"
##g

eax=00000000 ebx=00000014 ecx=0009a9ef edx=0000a9ef esi=00090bff edi=00090006
eip=0000272d esp=0000b230 ebp=0009b230 iopl=2 -- -- -- nv up ei ng nz na pe nc
cs=d02f ss=004f ds=a9ef es=be47 fs=150b gs=0000 cr2=01550000 cr3=001d9000
d02f:0000272e ca0800      retf      0008
Symbols linked (calc)
DelayHardError SYS3171: 4 string(s):
Pid 000b Tid 0001 Slot 002d HobMte 03be
C:\OS2TOOLS\MRFILEPM.EXE
c0000005
1a05272d
P1=00000008 P2=6d640000 P3=XXXXXXXX P4=XXXXXXXX
EAX=00000000 EBX=00000014 ECX=0009a9ef EDX=0000a9ef
ESI=00090bff EDI=00090006
DS=a9ef DSACC=00f3 DSLIM=00000fff
ES=be47 ESACC=00f3 ESLIM=000017f5
FS=150b FSACC=00f3 FSLIM=00000030
GS=0000 GSACC=**** GSLIM=*****
CS:EIP=d02f:0000272d CSACC=00df CSLIM=000054a3
SS:ESP=0000:0000b230 SSACC=**** SSLIM=*****
EBP=0009b230 FLG=00002386

DOSCALL1.DLL 0005:0000272d
```

This technique will successfully terminate an application. If **TRAPDUMP** is set appropriately then a system dump will be taken.

If **TRAPDUMP** is not correctly set for taking dumps, it may be dynamically modified from the debugging console. Symbol **DumpDevice** specifies the dump partition or drive letter (without the colon) and **DUMP\_ON** is a flag byte that take values 0, 1 or 2 to specify whether **TRAPDUMP** is **OFF**, **ON** or **R0** respectively. Use the **E command** to modify to these fields according to needs. For example, if we wish to set the equivalent of **TRAPDUMP R0,F** after system initialisation then the following command sequence would achieve this:

```
e dump_on 2
e dumpdevice "F"
```

When examining a dump taken by calling **RASRST**, directly or indirectly, using the **G command** then the registers at the time the Kernel Debugger was last entered can be found at label **\_RegSA**. The format of this save area is as follows.

Before fix pack 29 for Warp 3.0 and base Warp 4.0:

```
Offset Register mnemonic
+0      EAX
+4      EBX
+8      EXC
+c      EDX
+10     ESP
+14     EBP
+18     ESI
+1c     EDI
```

+20	ES
+22	SS
+24	DS
+26	FS
+28	GS
+2a	EIP
+2e	CS
+30	reserved
+34	EFLAGS
+38	MSW
+3c	GTD limit
+3e	GTD base
+42	reserved
+44	IDT limit
+46	IDT base
+4a	reserved
+4c	LDTR
+4e	TR
+50	CR2
+54	CR3
+58	DR0
+5c	DR1
+60	DR2
+64	DR3
+68	DR4
+6c	DR5
+70	DR6
+74	DR7
+78	reserved
+7c	TR6
+80	TR7

From fix pack 29 for Warp 3.0 and base Warp 4.0:

*Offset Register mnemonic*

+0	EAX
+4	EBX
+8	EXC
+c	EDX

+10	ESP
+14	EBP
+18	ESI
+1c	EDI
+20	ES
+22	SS
+24	DS
+26	FS
+28	GS
+2a	EIP
+2e	CS
+30	reserved
+34	EFLAGS
+38	MSW
+3c	GTD limit
+3e	GTD base
+42	reserved
+44	IDT limit
+46	IDT base
+4a	reserved
+4c	LDTR
+4e	TR
+50	CR2
+54	CR3
+58	CR4
+5c	DR0
+60	DR1
+64	DR2
+68	DR3
+6c	DR4
+70	DR5
+74	DR6
+78	DR7
+7c	reserved
+80	TR6
+84	TR7

---

## Virtual Memory Management Lock Trace

Virtual Memory Management implements a logging function that records successful attempts to lock and unlock memory pages.

Memory locking and unlocking is implemented by the Memory Management routines: **VMLockMem** and **VMUnlock**. This routine is available directly to all kernel components and indirectly to device drivers through:

**DevHlp\_Lock**

**DevHlp\_Unlock**

**DevHlp\_VMLock**

**DevHlp\_VMUnlock**

and to file system drivers through:

**MFSH\_Lock**

**MFSH\_Unlock**

The VM lock trace is activated by setting bit 0 of the VM log flag double-word to 1. The flag double word is located at symbol: **\_VMLogFlags**. Since no function is currently assigned to the other bit positions so the lock log may be effectively turned on by setting the byte a **\_VMLogFlags** to **0xff** as in the following example:

```
e _vmlogflags
%fff0127c 00.
ff
##g
L base fff32 size 2 flags 1 hob 16 hptda 3b9 ret fff3e551
L base 15e0 size 1 flags 4 hob 4a4 hptda 91 ret fff5a93c
L base 3f size 1 flags 4 hob 188 hptda 91 ret fff5a93c
U base 15e0 size 1 flags 4 hob 4a4 hptda 91 ret fff5a983
U base 3f size 1 flags 4 hob 188 hptda 91 ret fff5a983
L base 15e0 size 1 flags 4 hob 4a4 hptda 91 ret fff5a93c
L base 3f size 1 flags 4 hob 188 hptda 91 ret fff5a93c
U base 15e0 size 1 flags 4 hob 4a4 hptda 91 ret fff5a983
U base 3f size 1 flags 4 hob 188 hptda 91 ret fff5a983
L base fff35 size 3 flags 1 hob 16 hptda 4a4 ret fff3e551
L base fe79c size 4 flags 0 hob 3 hptda 380 ret fff49ec6
U base fe79c size 4 flags 0 hob 3 hptda 380 ret fff3d173
```

The fields displayed in each lock trace entry are formatted from the constituent parts of the corresponding lock handle. They are defined as follows:

- L** Indicates a lock request.
- U** Indicates an unlock request.
- base** The virtual page number (that is the high order 5 digits of the address) of the page(s) to be locked or unlocked.
- size** The number of pages being locked or unlocked
- flags** The following bit settings are defined:

Bit value	Description
0x01	Lock is a long-term
0x02	Verify lock call
0x04	Lock originated from a DevHlp

**hob**

The **hob** of the memory object whose pages are being locked or unlocked.

**hptda**

The **hptda** of the process that requested the memory lock or unlock.

**ret**

The return address from **VMLockMem**, that is, the address of the caller.

**Note:**

The return address is unfortunately of limited use since most calls to **VMLockMem** are made via a limited number of interface routines. In particular, **DevHlp** lock requests are made via **dhw\_VMLock** and **SegLockDM**. Unless one can trace in addition the **SS:ESP** on entry to **VMLock**, the lock trace alone will be insufficient to solve memory locking problem. One possible way of providing more information is to supplement the lock trace with following breakpoint commands:

```
##bp _vmunlock+1,"k ss:sp:g"
##bp _vmlockmem+1,"k ss:sp:g"
##g
0170:fff3e551 fff32d68 00001281 10000000 ffe0068f CodeLockProc + 7c
L base fff32 size 2 flags 1 hob 16 hptda 3b9 ret fff3e551
0170:fff5a93c 015f0000 0000000e 40000000 fe7958c6 _dhw_VMLock + dc
0170:fff3db40 40000000 015f0000 0000000e fe7958c6
0170:00000155 01550000 62d61a84 00000003 5ab40000
L base 15f0 size 1 flags 4 hob 5de hptda 91 ret fff5a93c
0170:fff5a93c 0003f198 0000000b 40000000 fe795be0 _dhw_VMLock + dc
0170:fff3db40 40000000 0003f198 0000000b fe795be0
0170:00000055 00550000 62d61a84 00000003 5ab40000
L base 3f size 1 flags 4 hob 196 hptda 91 ret fff5a93c
0170:fff5a983 fe7958c6 00002796 083082fc fe7958c6 _dhw_VMUnlock + 3a
0170:fff3db4c fe7958c6 00001af0 00001100 00000056
0170:00000003 5b030000 08300000 f2a40000 08489254
U base 15f0 size 1 flags 4 hob 5de hptda 91 ret fff5a983
0170:fff5a983 fe795be0 000008c6 0830823f fe795be0 _dhw_VMUnlock + 3a
0170:fff3db4c fe795be0 00001af0 00001100 0000ff56
0170:00000003 5b030000 08300000 f2a40000 08489254
U base 3f size 1 flags 4 hob 196 hptda 91 ret fff5a983
0170:fff5a93c 015f0000 00000dd6 40000000 fe7958c6 _dhw_VMLock + dc
0170:fff3db40 40000000 015f0000 00000dd6 fe7958c6
0170:00000155 01550000 62d61a84 00000003 5ab40000
L base 15f0 size 1 flags 4 hob 5de hptda 91 ret fff5a93c
0170:fff5a93c 0003f198 0000000b 40000000 fe795be0 _dhw_VMLock + dc
0170:fff3db40 40000000 0003f198 0000000b fe795be0
0170:00000055 00550000 62d61a84 00000003 5ab40000
L base 3f size 1 flags 4 hob 196 hptda 91 ret fff5a93c
0170:fff5a983 fe7958c6 00002796 083082fc fe7958c6 _dhw_VMUnlock + 3a
0170:fff3db4c fe7958c6 00001af0 00001100 0000ff56
0170:00000003 5b030000 08300000 f2a40000 08489254
U base 15f0 size 1 flags 4 hob 5de hptda 91 ret fff5a983
0170:fff5a983 fe795be0 000008c6 0830823f fe795be0 _dhw_VMUnlock + 3a
0170:fff3db4c fe795be0 00001af0 00001100 0000ff56
0170:00000003 5b030000 08300000 f2a40000 08489254
U base 3f size 1 flags 4 hob 196 hptda 91 ret fff5a983
0170:fff3e551 fff351dc 000022f5 10000000 ffe0053f CodeLockProc + 7c
L base fff35 size 3 flags 1 hob 16 hptda 3b9 ret fff3e551
0170:fff4a218 ffe0053f ffe0052b 00082006 00000000 _CodeLockHook + 2c
0170:fff42df7 ffffffff ffffffff 7b71ff40 7b71ff40 KMDispatchHook + a3
U base fff35 size 3 flags 1 hob 16 hptda 3b9 ret fff4a218
0170:fff4a218 ffe0068f ffe0067b 00082006 00000006 _CodeLockHook + 2c
0170:fff42df7 ffffffff ffffffff 7b71ff40 7b71ff40 KMDispatchHook + a3
U base fff32 size 2 flags 1 hob 16 hptda 3b9 ret fff4a218
```

Given that the **K** command rapidly loses synchronisation with the correct stack frame pointer one may have to resort to using:

```
##bp _vmunlock+1,"dw ss:sp 180:g"
##bp _vmlockmem+1,"dw ss:sp 180:g"
```

Refer to the Kernel Debugger [K command](#) and [BP command](#) for further information.

Related information on memory locking may be found under the description of the Kernel Debugger [.MO command](#).

The latest versions of OS/2 2.11 and OS/2 3.0 have implemented a new Kernel Debugger command that facilitates an alternative method for analysing memory locking problems. See the Kernel Debugger [.MK command](#) command for details.

-----

## Virtual Memory Management System Heap Validation

The system will perform additional validation of the kernel heap structures under the debug kernel if the byte at label: **\_vmkhGflags** is set to a non-zero value.

There is a noticeable performance overhead when this option is activated. Therefore it is recommended that it is only used when a heap corruption problem is suspected.

The system will validate the linkages between various heap structures. If an error is detected then an IPE is generated with one of the following messages:

VMKSH: Invalid hint pointers  
VMKSH: Invalid number of ksh descriptors  
VMKSH: Invalid number of ksh blocks  
Invalid heap block header at addr: ssss:00000000  
Preceding block at addr: ssss:00000000  
No preceding block

-----

## System Loader Logging Facility

The system [loader](#) provides optional logging and checking under the debug kernel. These optional facilities may be activated selectively by setting bits in the **\_LdrDebugFlags** flags double-word as follows:

### Note:

The flags described are those implemented in OS/2 Warp V3.0. Slightly different, similar messages are generated for earlier releases of OS/2.

### 0x00000001

This will cause the Loader to break into the debugger using an **INT 3** instruction if any of the following error conditions are detected:

Not enough memory  
Caching error  
Invalid Ordinal  
Procedure not found  
Bad EXE format  
Invalid segment number  
Invalid CALLGATE  
Network Disconnected

### 0x00000002

This will generate log entries when **LDRGetPage** exits with a non-zero return code. **LDRGetPage** is called to



demand load a page within a object of a load module. The message logged is of the following form:

```
ldrGP bad cr2=nnnnnnnn rc=nnnnnnnnnn
```

**cr2=** is the page fault address and **rc=** is the **LDRGetPage** return code.

#### 0x00000004

This generates log entries when **LDRGetPage** is called to demand load a page within a object of a load module. The message logged is of the form:

```
ldrGP cr2=nnnnnnnn hMTE=hhhh bno=oo  
name=pppppppppppppppppp
```

**cr2=** is the page fault address,

**hMTE=** is the module's [hnte](#),

**bno=** is page number with in the module

**name=** is the module's full name taken from the [SMTE](#).

#### 0x00000018

This switch causes log information to be generated when DLL modules are loaded and initialised. The following messages are logged:

```
ldrDLM entry - slot ssss ptda pppppppp  
ldrDLM name - slot ssss name nnnnnnnn  
ldrDLM free - slot ssss  
ldrDLM exit - slot ssss  
tk SD has-init slot=ssss  
tk SD no-init slot=ssss  
tk SD pre-inc slot=ssss cnest=nnnn  
tk IN pre-dec slot=ssss cnest=nnnn  
tk LIn slot=ssss cnest=nnnn
```

**slot** is the [thread slot](#) in which the DLL is being processed,

**ptda** is the address of the [PTDA](#) for this slot

**name** is the DLL module name

**cnest** Nesting counter for **TKLibiStartDispatch**

**ldrDLM entry** marks entry to **w\_loadmodule**, the **DosLoadModule** worker routine.

**ldrDLM name** marks the successful request for the DLL initialisation mutex semaphore (**ptda\_DLMsem (PTDA +0x4ac (H/R: +0x4a8))**).

**ldrDLM free** marks the release of the mutex semaphore. **Exit** marks the exiting of **w\_loadmodule**.

**ldrDLM exit** marks the exit from **w\_loadmodule**.

**tk SD** marks events in **TKLibiStartDispatch**.

**tk IN** and **tk LIn** Mark events in **TKLinilnitNextDLL**

#### 0x00000080

This switch requests import initialisation be recorded. Messages of the following format are generated:

```
lpi, Recording init hMTE=hhhh, flags1=ffffff, name=nnnnnnnnn  
lpi, Skipping init hMTE=hhhh, flags1=ffffff, name=nnnnnnnnn
```

```

lpi, Processing imports slot=ssss, module=nnnnnnnnn
lrm, Recording init hMTE=hhhh, flags1=ffffffff, name=nnnnnnnnn
lrm, Skipping init hMTE=hhhh, flags1=ffffffff, name=nnnnnnnnn

```

**hMTE** is the [module handle](#)

**flags1** are the flags [MTE flags](#) field. (See the [.LM command](#) for details).

**name** is the full module name taken from the [SMTE](#).

**module** is the full module name taken from the [SMTE](#).

**lpi, Recording init** Logs the processing of system DLL imports from the system DLL names table in EXE file loading.

**lpi, Skipping init** Logs system DLL names not imported in EXE file loading.

**lpi, Processing imports** Logs the processing of DLL initialisation as the result of imports being present in an EXE module.

**lrm, Recording init** Logs imported DLL initialisation being recorded.

**lrm, Skipping init** Logs imported DLLs skipping initialisation.

**0x00000100**

Logs when the loader cannot load an object at the compiler/linker designated base address. The message logged appears as:

```

Cannot load nnnnnnnnn at the requested base address

```

where *nnnnnnnn* is the module name.

**0x00000800**

Logs the processing of the DLL import tree. The following messages appear:

```

lpi, Processing imports slot=ssss, module=nnnnnnnnn
ldr walking tree hMTE=hhhh, name=nnnnnnnnn
ldr walking tree going down
ldr walking tree going up

```

**lpi, Processing imports** marks the initiation of the process for slot *ssss* and module *nnnnnnnn*.

**ldr walking tree hMTE=hhhh, name=nnnnnnnn** marks the processing of an imported DLL, whose handle is *hhhh* and name is *nnnnnnnn*

**ldr walking tree going up** marks a backward progression through the import tree.

**ldr walking tree going down** marks a forward progression through the import tree.

-----

## Example Loader Log

The following is an example of a loader log where all logging options have been activated. This illustrates the loader activity recorded when the **FAXWORK.EXE** icon was clicked on:

```
lpi Processing imports slot=0022, module=H:\FAXWORKS\FAXWORKS.EXE
ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE
ldr walking tree going down
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE
ldr walking tree going down
ldr walking tree hMTE=02b8, name=H:\OS2\DLL\PMSP.L.DLL
ldr walking tree going down
ldr walking tree hMTE=0104, name=H:\OS2\DLL\MSG.DLL
ldr walking tree going up
ldr walking tree hMTE=02b8, name=H:\OS2\DLL\PMSP.L.DLL
ldr walking tree going down
ldr walking tree hMTE=02be, name=H:\OS2\DLL\SPL1B.DLL
ldr walking tree going up
ldr walking tree hMTE=02b8, name=H:\OS2\DLL\PMSP.L.DLL
ldr walking tree going up
ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE
ldr walking tree going down
ldr walking tree hMTE=0419, name=H:\OS2\DLL\HELPMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE
ldr walking tree going down
ldr walking tree hMTE=02ac, name=H:\OS2\DLL\PMDRAG.DLL
ldr walking tree going down
ldr walking tree hMTE=02a3, name=H:\OS2\DLL\PMCTLS.DLL
ldr walking tree going up
ldr walking tree hMTE=02ac, name=H:\OS2\DLL\PMDRAG.DLL
ldr walking tree going up
ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE
ldr walking tree going down
ldr walking tree hMTE=0279, name=H:\OS2\DLL\PMWP.DLL
ldr walking tree going down
ldr walking tree hMTE=029d, name=H:\OS2\DLL\IMP.DLL
ldr walking tree going up
ldr walking tree hMTE=0279, name=H:\OS2\DLL\PMWP.DLL
ldr walking tree going down
ldr walking tree hMTE=02a8, name=H:\OS2\DLL\SEAMLESS.DLL
ldr walking tree going down
ldr walking tree hMTE=02b1, name=H:\OS2\DLL\PMVIOP.DLL
ldr walking tree going up
ldr walking tree hMTE=02a8, name=H:\OS2\DLL\SEAMLESS.DLL
ldr walking tree going up
ldr walking tree hMTE=0279, name=H:\OS2\DLL\PMWP.DLL
ldr walking tree going down
```

ldr walking tree hMTE=02ad, name=H:\OS2\DLL\SOM.DLL  
ldr walking tree going up  
ldr walking tree hMTE=0279, name=H:\OS2\DLL\PMWP.DLL  
ldr walking tree going up  
ldr walking tree hMTE=05e1, name=H:\FAXWORKS\FAXWORKS.EXE  
ldr walking tree going up  
lrm, Skipping init hMTE=05e1, flags1=20903150, name=H:\FAXWORKS\FAXWORKS.EXE  
lrm, Skipping init hMTE=0279, flags1=e498b394, name=H:\OS2\DLL\PMWP.DLL  
lrm, Skipping init hMTE=02ad, flags1=e498b396, name=H:\OS2\DLL\SOM.DLL  
lrm, Recording init hMTE=02a8, flags1=e098b395, name=H:\OS2\DLL\SEAMLESS.DLL  
lrm, Skipping init hMTE=02b1, flags1=a498b395, name=H:\OS2\DLL\PMVIOP.DLL  
lrm, Skipping init hMTE=029d, flags1=2098b398, name=H:\OS2\DLL\IMP.DLL  
lrm, Skipping init hMTE=02ac, flags1=a498b388, name=H:\OS2\DLL\PMDRAG.DLL  
lrm, Skipping init hMTE=02a3, flags1=e498b394, name=H:\OS2\DLL\PMCTLS.DLL  
lrm, Skipping init hMTE=0419, flags1=a098b39a, name=H:\OS2\DLL\HELPMGR.DLL  
lrm, Skipping init hMTE=02b8, flags1=e498b394, name=H:\OS2\DLL\PMSPL.DLL  
lrm, Skipping init hMTE=02be, flags1=a098b398, name=H:\OS2\DLL\SPL1B.DLL  
lrm, Skipping init hMTE=0104, flags1=2098b388, name=H:\OS2\DLL\MSG.DLL  
lrm, Skipping init hMTE=029a, flags1=2098b388, name=H:\OS2\DLL\PMWIN.DLL  
lrm, Skipping init hMTE=0281, flags1=e498b394, name=H:\OS2\DLL\PMMERGE.DLL  
lrm, Skipping init hMTE=0111, flags1=2098b388, name=H:\OS2\DLL\SESMGR.DLL  
lrm, Skipping init hMTE=029c, flags1=2098b388, name=H:\OS2\DLL\PMSHAPI.DLL  
lrm, Skipping init hMTE=0262, flags1=2098b388, name=H:\OS2\DLL\NLS.DLL  
lrm, Skipping init hMTE=01e5, flags1=2098b388, name=H:\OS2\DLL\VIOCALLS.DLL  
lrm, Skipping init hMTE=029b, flags1=2098b388, name=H:\OS2\DLL\MOUCALLS.DLL  
lrm, Recording init hMTE=0293, flags1=e498b394, name=H:\OS2\DLL\PMGPI.DLL  
lpi, Recording init hMTE=0279, flags1=e498b394, name=H:\OS2\DLL\PMWP.DLL  
lpi, Recording init hMTE=02ad, flags1=e498b396, name=H:\OS2\DLL\SOM.DLL  
lpi, Recording init hMTE=02b1, flags1=a498b395, name=H:\OS2\DLL\PMVIOP.DLL  
lpi, Recording init hMTE=02a3, flags1=e498b394, name=H:\OS2\DLL\PMCTLS.DLL  
lpi, Recording init hMTE=02ac, flags1=a498b388, name=H:\OS2\DLL\PMDRAG.DLL  
lpi, Recording init hMTE=02b8, flags1=e498b394, name=H:\OS2\DLL\PMSPL.DLL  
lpi, Recording init hMTE=0281, flags1=e498b394, name=H:\OS2\DLL\PMMERGE.DLL  
lpi, Recording init hMTE=00f2, flags1=8498b794, name=H:\OS2\DLL\DOSCALL1.DLL  
ldrGP cr2=ffe3b000 hMTE=5e1 bno=85  
    name = H:\FAXWORKS\FAXWORKS.EXE  
tk SD has-init slot=22  
ldrGP cr2=13fa0000 hMTE=f2 bno=2b  
    name = H:\OS2\DLL\DOSCALL1.DLL  
ldrGP cr2=13fa0000 hMTE=f2 bno=2c  
    name = H:\OS2\DLL\DOSCALL1.DLL  
ldrGP cr2=13fc0000 hMTE=f2 bno=2e  
    name = H:\OS2\DLL\DOSCALL1.DLL  
ldrGP cr2=13e30000 hMTE=281 bno=106  
    name = H:\OS2\DLL\PMMERGE.DLL  
ldrDLM entry - slot 36 ptda ab99a000  
ldrDLM name - slot 36 name H:\OS2\DLL\PMATM.DLL  
lpi Processing imports slot=0036, module=H:\OS2\DLL\PMATM.DLL  
ldr walking tree hMTE=0354, name=H:\OS2\DLL\PMATM.DLL  
ldr walking tree going down  
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL  
ldr walking tree going down  
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL  
ldr walking tree going down  
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL  
ldr walking tree going up  
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL  
ldr walking tree going down  
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL  
ldr walking tree going up  
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL  
ldr walking tree going down  
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL  
ldr walking tree going up  
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL  
ldr walking tree going down  
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL  
ldr walking tree going up  
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL  
ldr walking tree going down  
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL  
ldr walking tree going up  
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL  
ldr walking tree going down  
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL  
ldr walking tree going up  
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL  
ldr walking tree going up  
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL

```

ldr walking tree going up
ldr walking tree hMTE=0354, name=H:\OS2\DLL\PMATM.DLL
ldr walking tree going up
lrm, Recording init hMTE=0354, flags1=6498b3c5, name=H:\OS2\DLL\PMATM.DLL
lrm, Skipping init hMTE=029c, flags1=2098b388, name=H:\OS2\DLL\PMSHAPI.DLL
tk SD has-init slot=36
tk SD pre-inc slot=36 cnest=1
ldrDLM free - slot 36
ldrDLM exit - slot 36
tk LIn slot=36 cnest=1
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name DISPLAY
lpi Processing imports slot=0036, module=H:\OS2\DLL\DISPLAY.DLL
ldr walking tree hMTE=034b, name=H:\OS2\DLL\DISPLAY.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=034b, name=H:\OS2\DLL\DISPLAY.DLL
ldr walking tree going up
lrm, Recording init hMTE=034b, flags1=2498b394, name=H:\OS2\DLL\DISPLAY.DLL
tk SD has-init slot=36
tk SD pre-inc slot=36 cnest=1
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=13ef0000 hMTE=34b bno=6
name = H:\OS2\DLL\DISPLAY.DLL
tk LIn slot=36 cnest=1
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name IBMS332
lpi Processing imports slot=0036, module=H:\OS2\DLL\IBMS332.DLL
ldr walking tree hMTE=0362, name=H:\OS2\DLL\IBMS332.DLL
ldr walking tree going down
ldr walking tree hMTE=0368, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL

```

```
ldr walking tree going down
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=0368, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going up
ldr walking tree hMTE=0362, name=H:\OS2\DLL\IBMS332.DLL
ldr walking tree going up
lrm, Skipping init hMTE=0362, flags1=2098b398, name=H:\OS2\DLL\IBMS332.DLL
lrm, Skipping init hMTE=0368, flags1=2098b388, name=H:\OS2\DLL\PMGRE.DLL
tk SD no-init slot=36
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name H:\OS2\DLL\IBMS332.DLL
tk SD no-init slot=36
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name dsPRES
lpi Processing imports slot=0036, module=H:\OS2\DLL\DSPRES.DLL
ldr walking tree hMTE=036a, name=H:\OS2\DLL\DSPRES.DLL
ldr walking tree going up
lrm, Skipping init hMTE=036a, flags1=2098b388, name=H:\OS2\DLL\DSPRES.DLL
tk SD no-init slot=36
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name H:\OS2\DLL\COMETDLL.DLL
lpi Processing imports slot=0036, module=H:\OS2\DLL\COMETDLL.DLL
ldr walking tree hMTE=037e, name=H:\OS2\DLL\COMETDLL.DLL
ldr walking tree going down
ldr walking tree hMTE=0368, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=0368, name=H:\OS2\DLL\PMGRE.DLL
ldr walking tree going up
ldr walking tree hMTE=037e, name=H:\OS2\DLL\COMETDLL.DLL
ldr walking tree going down
ldr walking tree hMTE=0104, name=H:\OS2\DLL\MSG.DLL
ldr walking tree going up
```

```

ldr walking tree hMTE=037e, name=H:\OS2\DLL\COMETDLL.DLL
ldr walking tree going up
lrm, Recording init hMTE=037e, flags1=e098b396, name=H:\OS2\DLL\COMETDLL.DLL
lrm, Skipping init hMTE=0104, flags1=2098b388, name=H:\OS2\DLL\MSG.DLL
lrm, Skipping init hMTE=0368, flags1=2098b388, name=H:\OS2\DLL\PMGRE.DLL
lrm, Skipping init hMTE=029a, flags1=2098b388, name=H:\OS2\DLL\PMWIN.DLL
tk SD has-init slot=36
tk SD pre-inc slot=36 cnest=1
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=13b30000 hMTE=37e bno=7
      name = H:\OS2\DLL\COMETDLL.DLL
tk LIn slot=36 cnest=1
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name PMSPL
tk SD no-init slot=36
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=13d90000 hMTE=2b8 bno=31
      name = H:\OS2\DLL\PMSPL.DLL
ldrGP cr2=13940000 hMTE=2a3 bno=7b
      name = H:\OS2\DLL\PMCTLS.DLL
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name PMSDMRI
lpi Processing imports slot=0036, module=H:\OS2\DLL\PMSDMRI.DLL
ldr walking tree hMTE=02c6, name=H:\OS2\DLL\PMSDMRI.DLL
ldr walking tree going up
lrm, Skipping init hMTE=02c6, flags1=2098b388, name=H:\OS2\DLL\PMSDMRI.DLL
tk SD no-init slot=36
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=13c65000 hMTE=2ad bno=1c
      name = H:\OS2\DLL\SOM.DLL
ldrGP cr2=13f60000 hMTE=279 bno=bf
      name = H:\OS2\DLL\PMWP.DLL
ldrGP cr2=13f40000 hMTE=279 bno=ba
      name = H:\OS2\DLL\PMWP.DLL
ldrGP cr2=13f7d000 hMTE=279 bno=cf
      name = H:\OS2\DLL\PMWP.DLL
tk LIn slot=36 cnest=0
ldrGP cr2=47000 hMTE=5e1 bno=38
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=e6000 hMTE=5e1 bno=55
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=54000 hMTE=5e1 bno=45
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4a000 hMTE=5e1 bno=3b
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=ec000 hMTE=5e1 bno=5b
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=f0000 hMTE=5e1 bno=5f
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=48000 hMTE=5e1 bno=39
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=52000 hMTE=5e1 bno=43
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=e8000 hMTE=5e1 bno=57
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=f8000 hMTE=5e1 bno=67
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=e7000 hMTE=5e1 bno=56
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=f1000 hMTE=5e1 bno=60
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=ee000 hMTE=5e1 bno=5d
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=49000 hMTE=5e1 bno=3a
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=ed000 hMTE=5e1 bno=5c
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=13000 hMTE=5e1 bno=4
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=115000 hMTE=5e1 bno=84
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=32000 hMTE=5e1 bno=23
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=e1000 hMTE=5e1 bno=50
      name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=33000 hMTE=5e1 bno=24

```

```

        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4c000 hMTE=5e1 bno=3d
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4b000 hMTE=5e1 bno=3c
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name H:\FAXWORKS\FX044.LOL
lpi Processing imports slot=0036, module=H:\FAXWORKS\FX044.LOL
ldr walking tree hMTE=060b, name=H:\FAXWORKS\FX044.LOL
ldr walking tree going up
lrm, Skipping init hMTE=060b, flags1=2098b1c8, name=H:\FAXWORKS\FX044.LOL
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name SND
lpi Processing imports slot=0036, module=H:\MMOS2\DLL\SND.DLL
ldr walking tree hMTE=00fe, name=H:\MMOS2\DLL\SND.DLL
ldr walking tree going down
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=00fe, name=H:\MMOS2\DLL\SND.DLL
ldr walking tree going down
ldr walking tree hMTE=0104, name=H:\OS2\DLL\MSG.DLL
ldr walking tree going up
ldr walking tree hMTE=00fe, name=H:\MMOS2\DLL\SND.DLL
ldr walking tree going up
lrm, Recording init hMTE=00fe, flags1=6098b396, name=H:\MMOS2\DLL\SND.DLL
lrm, Skipping init hMTE=0104, flags1=2098b388, name=H:\OS2\DLL\MSG.DLL
lrm, Skipping init hMTE=029a, flags1=2098b388, name=H:\OS2\DLL\PMWIN.DLL
lrm, Skipping init hMTE=029c, flags1=2098b388, name=H:\OS2\DLL\PMSHAPI.DLL
lrm, Skipping init hMTE=0262, flags1=2098b388, name=H:\OS2\DLL\NLS.DLL
tk SD has-init slot=36
tk SD pre-inc slot=36 cnest=1
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=13310000 hMTE=fe bno=12
        name = H:\MMOS2\DLL\SND.DLL
ldrGP cr2=13311000 hMTE=fe bno=13
        name = H:\MMOS2\DLL\SND.DLL
tk LIn slot=36 cnest=1
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name PMCTLS
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=45000 hMTE=5e1 bno=36
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4e000 hMTE=5e1 bno=3f
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178f5000 hMTE=60b bno=6
        name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=e2000 hMTE=5e1 bno=51

```



```

name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178f6000 hMTE=60b bno=7
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178f5000 hMTE=60b bno=6
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=80000 hMTE=5e1 bno=49
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=3c000 hMTE=5e1 bno=2d
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=39000 hMTE=5e1 bno=2a
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=fc000 hMTE=5e1 bno=6b
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=f9000 hMTE=5e1 bno=68
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=fa000 hMTE=5e1 bno=69
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=e4000 hMTE=5e1 bno=53
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=e5000 hMTE=5e1 bno=54
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=fb000 hMTE=5e1 bno=6a
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=2a000 hMTE=5e1 bno=1b
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=2c000 hMTE=5e1 bno=1d
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=27000 hMTE=5e1 bno=18
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=4d000 hMTE=5e1 bno=3e
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=55000 hMTE=5e1 bno=46
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=e0000 hMTE=5e1 bno=4f
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=10b000 hMTE=5e1 bno=7a
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=10c000 hMTE=5e1 bno=7b
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=f7000 hMTE=5e1 bno=66
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=1f000 hMTE=5e1 bno=10
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=20000 hMTE=5e1 bno=11
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=ef000 hMTE=5e1 bno=5e
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=25000 hMTE=5e1 bno=16
name = H:\FAXWORKS\FAXWORKS.EXE
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name H:\FAXWORKS\Fax.adp
lpi Processing imports slot=0036, module=H:\FAXWORKS\FAX.ADP
ldr walking tree hMTE=0618, name=H:\FAXWORKS\FAX.ADP
ldr walking tree going down
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going down
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0293, name=H:\OS2\DLL\PMGPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029b, name=H:\OS2\DLL\MOUCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=01e5, name=H:\OS2\DLL\VIOCALLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0262, name=H:\OS2\DLL\NLS.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=029c, name=H:\OS2\DLL\PMSHAPI.DLL
ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going down
ldr walking tree hMTE=0111, name=H:\OS2\DLL\SESMGR.DLL

```

```

ldr walking tree going up
ldr walking tree hMTE=0281, name=H:\OS2\DLL\PMMERGE.DLL
ldr walking tree going up
ldr walking tree hMTE=029a, name=H:\OS2\DLL\PMWIN.DLL
ldr walking tree going up
ldr walking tree hMTE=0618, name=H:\FAXWORKS\FAX.ADP
ldr walking tree going down
ldr walking tree hMTE=0104, name=H:\OS2\DLL\MSG.DLL
ldr walking tree going up
ldr walking tree hMTE=0618, name=H:\FAXWORKS\FAX.ADP
ldr walking tree going up
lrm, Recording init hMTE=0618, flags1=6090b1c6, name=H:\FAXWORKS\FAX.ADP
lrm, Skipping init hMTE=0104, flags1=2098b388, name=H:\OS2\DLL\MSG.DLL
lrm, Skipping init hMTE=029a, flags1=2098b388, name=H:\OS2\DLL\PMWIN.DLL
lrm, Skipping init hMTE=029c, flags1=2098b388, name=H:\OS2\DLL\PMSHAPI.DLL
lrm, Skipping init hMTE=0262, flags1=2098b388, name=H:\OS2\DLL\NLS.DLL
tk SD has-init slot=36
tk SD pre-inc slot=36 cnest=1
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=178de000 hMTE=618 bno=f
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=178df000 hMTE=618 bno=10
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=10e72000 hMTE=618 bno=1c
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=178e6000 hMTE=618 bno=17
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=178e0000 hMTE=618 bno=11
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=10e73000 hMTE=618 bno=1d
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=10e74000 hMTE=618 bno=1e
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=178e5000 hMTE=618 bno=16
    name = H:\FAXWORKS\FAX.ADP
tk LIn slot=36 cnest=1
ldrGP cr2=178d1000 hMTE=618 bno=2
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=13e51000 hMTE=281 bno=118
    name = H:\OS2\DLL\PMMERGE.DLL
ldrGP cr2=178db000 hMTE=618 bno=c
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=10e70000 hMTE=618 bno=1a
    name = H:\FAXWORKS\FAX.ADP
ldrGP cr2=114000 hMTE=5e1 bno=83
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=26000 hMTE=5e1 bno=17
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=14000 hMTE=5e1 bno=5
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=13f7e000 hMTE=279 bno=d0
    name = H:\OS2\DLL\PMWP.DLL
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name COMETDLL
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=178c2000 hMTE=5e1 bno=91
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c0000 hMTE=5e1 bno=86
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=11000 hMTE=5e1 bno=2
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=3b000 hMTE=5e1 bno=2c
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c6000 hMTE=60b bno=7
    name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c7000 hMTE=60b bno=8
    name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c0000 hMTE=60b bno=1
    name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c0000 hMTE=60b bno=1
    name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c0000 hMTE=60b bno=1
    name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c0000 hMTE=5e1 bno=86
    name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c1000 hMTE=5e1 bno=87
    name = H:\FAXWORKS\FAXWORKS.EXE

```

```
ldrGP cr2=178c1000 hMTE=5e1 bno=87
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c1000 hMTE=5e1 bno=87
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=f6000 hMTE=5e1 bno=65
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c1000 hMTE=5e1 bno=87
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c1000 hMTE=5e1 bno=87
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c1000 hMTE=5e1 bno=87
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c2000 hMTE=5e1 bno=88
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c6000 hMTE=60b bno=7
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c7000 hMTE=60b bno=8
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=13c80000 hMTE=419 bno=34
name = H:\OS2\DLL\HELPMGR.DLL
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name HPMGRMRI
lpi Processing imports slot=0036, module=H:\OS2\DLL\HPMGRMRI.DLL
ldr walking tree hMTE=062a, name=H:\OS2\DLL\HPMGRMRI.DLL
ldr walking tree going up
lrm, Skipping init hMTE=062a, flags1=2098b18a, name=H:\OS2\DLL\HPMGRMRI.DLL
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=178c7000 hMTE=62a bno=8
name = H:\OS2\DLL\HPMGRMRI.DLL
ldrGP cr2=10000 hMTE=5e1 bno=1
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=113000 hMTE=5e1 bno=82
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=21000 hMTE=5e1 bno=12
name = H:\FAXWORKS\FAXWORKS.EXE
ldrDLM entry - slot 36 ptda ab99a000
ldrDLM name - slot 36 name H:\OS2\DLL\HELV.FON
lpi Processing imports slot=0036, module=H:\OS2\DLL\HELV.FON
ldr walking tree hMTE=035e, name=H:\OS2\DLL\HELV.FON
ldr walking tree going up
lrm, Skipping init hMTE=035e, flags1=2098b3c8, name=H:\OS2\DLL\HELV.FON
ldrDLM free - slot 36
ldrDLM exit - slot 36
ldrGP cr2=28000 hMTE=5e1 bno=19
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c2000 hMTE=60b bno=3
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=42000 hMTE=5e1 bno=33
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=44000 hMTE=5e1 bno=35
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=41000 hMTE=5e1 bno=32
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c0000 hMTE=60b bno=1
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c6000 hMTE=60b bno=7
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=178c7000 hMTE=60b bno=8
name = H:\FAXWORKS\FX044.LOL
ldrGP cr2=43000 hMTE=5e1 bno=34
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=40000 hMTE=5e1 bno=31
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=29000 hMTE=5e1 bno=1a
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=53000 hMTE=5e1 bno=44
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=3d000 hMTE=5e1 bno=2e
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=112000 hMTE=5e1 bno=81
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=12000 hMTE=5e1 bno=3
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c2000 hMTE=5e1 bno=91
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c2000 hMTE=5e1 bno=91
name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP cr2=178c0000 hMTE=5e1 bno=8f
```

```

        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c0000 hMTE=5e1 bno=8f
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c0000 hMTE=5e1 bno=8f
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c0000 hMTE=5e1 bno=8f
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c0000 hMTE=5e1 bno=8f
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c0000 hMTE=5e1 bno=8f
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c0000 hMTE=5e1 bno=8f
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c1000 hMTE=5e1 bno=90
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c2000 hMTE=5e1 bno=91
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c1000 hMTE=5e1 bno=90
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c1000 hMTE=5e1 bno=90
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c1000 hMTE=5e1 bno=90
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c1000 hMTE=5e1 bno=90
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c1000 hMTE=5e1 bno=90
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c1000 hMTE=5e1 bno=90
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c2000 hMTE=5e1 bno=91
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c2000 hMTE=5e1 bno=91
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178c6000 hMTE=60b bno=7
        name = H:\FAXWORKS\FX044.LOL
ldrGP  cr2=178c6000 hMTE=60b bno=7
        name = H:\FAXWORKS\FX044.LOL
ldrGP  cr2=22000 hMTE=5e1 bno=13
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=24000 hMTE=5e1 bno=15
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=37000 hMTE=5e1 bno=28
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=23000 hMTE=5e1 bno=14
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=13e52000 hMTE=281 bno=119
        name = H:\OS2\DLL\PMMERGE.DLL
ldrGP  cr2=13e53000 hMTE=281 bno=11a
        name = H:\OS2\DLL\PMMERGE.DLL
ldrGP  cr2=13e54000 hMTE=281 bno=11b
        name = H:\OS2\DLL\PMMERGE.DLL

```

The following shows the loader sequence when **FAXWORKS.EXE** is terminated:

```

ldrGP  cr2=178c6000 hMTE=60b bno=7
        name = H:\FAXWORKS\FX044.LOL
ldrGP  cr2=178c7000 hMTE=60b bno=8
        name = H:\FAXWORKS\FX044.LOL
ldrGP  cr2=178c6000 hMTE=60b bno=7
        name = H:\FAXWORKS\FX044.LOL
ldrGP  cr2=178c6000 hMTE=60b bno=7
        name = H:\FAXWORKS\FX044.LOL
ldrGP  cr2=50000 hMTE=5e1 bno=41
        name = H:\FAXWORKS\FAXWORKS.EXE
ldrGP  cr2=178e4000 hMTE=618 bno=15
        name = H:\FAXWORKS\FAX.ADP

```

-----

## DosDebug Logging Facility

The kernel worker routines for the **DosDebug** API implement a number of logging functions for use in debugging errors in **DosDebug** itself. These are activated by setting bits in the double-word at symbol: **\_DBGbugbug**.

The following flags bits are defined:

<b>0x01000000</b>	Display input to DosDebug
<b>0x02000000</b>	Display output from DosDebug
<b>0x00000010</b>	Display exceptions in DosDebug processing.
<b>0x10000000</b>	Display execution flow in debugger processing.
<b>0x20000000</b>	Display execution flow in debuggee processing.
<b>0x40000000</b>	Display execution flow in watchpoint and debug register processing.

-----

## DosPTrace Logging Facility

The kernel worker routines for the **DosPTrace** API implement a number of logging functions for use in debugging errors in **DosPTrace** itself. These are activated by setting bits in the double-word at symbol: **\_PTbugbug**.

**Note:**

**DosPTrace** internally thanks to **DosDebug** therefore [DosDebug Logging Facility](#) may be a useful diagnostic aid with **DosPTrace**.

The following flags bits are defined:

<b>0x01000000</b>	Display output buffer set-up passed to user.
<b>0x02000000</b>	Display input buffer set-up passed from user.
<b>0x04000000</b>	Display conversion routine flow.
<b>0x08000000</b>	Display alias conversion routine flow.
<b>0x10000000</b>	Display input and output return codes only.
<b>0x20000000</b>	Display processing of notifications from DosDebug.
<b>0x00000001</b>	Display floating point information. processing.

-----

## Kernel Debugger Breakpoints

The [break-point command set](#) of the Kernel Debugger provides a mechanism for intercepting the execution of code through a particular

path. For debugging application programs, break-points are generally required within the application itself or on call to or return from one or more system APIs.

Each system API results either in a call to a system DLL or to the Kernel through a CallGate. The name of a system interface that is called when an application uses an API is either identical to the API name or may be determined from one of the following conventions:

Dosl *name*                      Kernel Callgate name corresponding to API Dos *name*.

Dos32 <i>name</i>	DOSCALL1 32-bit entry point corresponding to API Dos <i>name</i> .
-------------------	--

Dos16 <i>name</i>	DOSCALL1 16-bit entry point corresponding to API Dos <i>name</i> .
-------------------	--

Other system DLLs such as PMWIN.DLL, PMMERGE.DLL, etc. adopt similar conventions, for example API **WinCreateWindow** calls **Win32CreateWindow** in PMMERGE.DLL.

In nearly all cases the system entry points have corresponding system tracepoints with the entry point name prefixed with either *pre* or *post*. Thus the [System Tracepoints Reference](#) provides a comprehensive source for deriving API related break-points.

Physical Device Driver helper routines pass through a common router, then to specific worker routines. Worker entry point names generally adhere to the following convention:

`DosHlp_name` worker routine `dh_name`.

Virtual Device Driver helper routines have entry points in the kernel with identical names (folded to uppercase) to the helper name.

File System Driver and Mini-File System Driver helper routines have entry points in the kernel with identical names to the helper name.

In addition to API and Driver Helper related break-points, the following system labels may also prove useful when intercepting errors or program initiation:

\_tkSchedNext

This routine is called when a new thread is selected for scheduling. The out-going thread slot number is recorded in variable **Tasknumber**.

**\_tkSchedNext** exits from one of two points:

SchedNextRet	A new thread slot is selected.
--------------	--------------------------------

SchedNextRet2                      The same thread slot is selected.

These labels maybe used to obtain a trace of dispatching activity. This is particularly useful when trying to establish the scope of hang conditions.

The following example illustrates how to obtain a trace of dispatched tasks using this break-point.

```
##bp _tkschednext, ".p #ig"
```

##g

Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0033#	0019	0000	0019	0001	blk	081e	7b98c000	7bb4b288	7bb2d394	1bf8	10	wkstahlp
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0034#	0018	0000	0018	0002	run	021f	7b98e000	7bb4aa5c	7bb2d548	1ea8	10	wksta
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0034#	0018	0000	0018	0002	blk	021f	7b98e000	7bb4aa5c	7bb2d548	1f00	10	wksta
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0038#	0018	0000	0018	0003	blk	0200	7b996000	7bb4aa5c	7bb2dc18	1eb8	10	wksta
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*000b#	0004	0000	0004	0001	blk	080b	7b93c000	7bb45078	7bb28f74	1cf0	00	land11
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell
Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
*0008#	0006	0001	0006	0001	blk	0500	7b936000	7bb460d0	7bb28a58	1eb8	01	pmshell

**Note:**

The status shows as blocked since **\_tkSchedNext** has been called because the current thread is giving up its time-slice.

**DosLibIDisp**

This API is called to initiate DLL initialisation whenever a new module is loaded into memory. Since this is called for every .EXE at load time, in the context of the new process and thread, it provides an excellent breakpoint for intercepting the loading of a new module in a new process.

When DosLibIDisp receives control, the **MTE**, **SMTE** have been created and the program module has been loaded. From the SMTE we can determine the entry point of the new module and thus set a breakpoint on this address.

The following example illustrates how to set a breakpoint on entry to a new module.

```
>> Add breakpoint at DosLibIDisp, then start CMD.EXE

##bp doslibDisp
##g
eax=00000000 ebx=000029f4 ecx=00000010 edx=00000014 esi=00000bc8 edi=00000c0a
eip=00000294 esp=0000773c ebp=00007752 iopl=2 -- -- -- nv up ei pl nz na po nc
cs=ffd7 ss=001f ds=ffa7 es=ffaf fs=150b gs=0000 cr2=1fc70490 cr3=001d0000
doscall11:CODE16_GROUP:DOSLIBIDISP:
ffd7:00000294 b80100          mov     ax,0001          ;br0

##.p#
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
*0044# 002c 0006 002c 0001 run  0400 7b9ae000 7bb4fc14 7bb2f088 1f48 19 cmd

>> The hmte for the current process is found in the PTDA at
>> ptda_module

##dw ptda_module l1
0030:0000ffaa 03a1

##.lmo 3a1
hmte=03a1 pmte=%fe97ebe4 mflags=84903152 c:\os2\cmd.exe
obj  vsize  vbase  flags  ipagemap  cpagemap  hob  sel
0001 0000c6a8 00010000 80001025 00000001 0000000d 03a0 000f r-x shr alias
0002 00007efa 00020000 80001025 0000000e 00000008 03a2 0017 r-x shr alias
0003 00009730 00030000 80001043 00000016 00000002 0000 001f rw- prel alias

>> Now dump the MTE and SMTE, whose address is at MTE+0x4

##dd %fe97ebe4 l8
%fe97ebe4 03a10002 fd4341d0 fe97ec1c fe9a143c
%fe97ebf4 84903152 00000007 00060050 fe908e74

##dd %fd4341d0
%fd4341d0 00000017 00000002 000044fa 00000003
%fd4341e0 00007790 00000009 000005d9 fd434261
%fd4341f0 00000003 fd4342a9 00000a00 00000000
%fd434200 00000000 fd434361 fd434368 fd434369
%fd434210 fd4343c9 fd4348f1 fd434924 00000a00
%fd434220 00000000 00000000 00000003 00000000
%fd434230 00000000 00001fa0 fd434252 00000000
%fd434240 00000000 00003f40 00000000 0000000e

>> SMTE+0x4 is the entry point object number
>> SMTE+0x8 is the entry point offset offset
>> For CMD.EXE this is 2:44fa
>> Since object 2 starts at %20000, we can define a breakpoint on
>> entry to CMD.EXE at %20000+44fa

##bp %00020000 +44fa

##bl
0 e ffd7:00000294 [DOSLIBIDISP]
1 e %000244fa [__astart]

>> Disable BP 0 since DosLibIDisp is called for every DLL that will be
>> initialised in the new process.

##bd 0
```

```

##g
eax=00000027 ebx=00000491 ecx=00009730 edx=0000f834 esi=00001fa0 edi=000003a1
eip=000044fa esp=00007790 ebp=00000000 iopl=2 -- -- -- nv up ei pl nz na po nc
cs=0017  ss=001f  ds=001f  es=0000  fs=150b  gs=0000   cr2=00063ffe  cr3=001d0000
cmd: _TEXT3: __astart:
0017:000044fa fc          cld                      ;br2

```

## VMLockMem

This breakpoint is on entry to the memory locking subroutine of Virtual Memory Management. It may be used in conjunction with the [VM Lock Trace](#)

## \_XCPTBuildR3DispatcherStack

This routine is called whenever a process fatal exception is generated by the kernel, regardless of whether exception handlers are registered. It therefore makes a stronger method than **VSF \*** for intercepting fatal user exceptions.

Exception management and how to intercept exceptions is discussed in more detail in [Trap and Exception Processing](#).

## Dos32R3ExceptionDispatcher

This entry-point is DOSCALL1.DLL is called by the kernel to process all user exception handlers. A breakpoint on the label allows one to intercept user exceptions before the user context is modified by any user exception handlers. On entry **ESP+0x4** contains the trap number, **ESP+0x8** points to the [exception report record](#) and **ESP+0xc** points to the [exception context record](#).

## \_xcptrR3ExceptionDispatcher.

This routine is called from **Dos32R3ExceptionDispatcher** to process each of the user exception handlers. It does this by locating exception registration records from the [TIB](#) at +0x0.

On entry to the Ring 3 Exception Dispatcher, **ESP+0x4** and **EXP+0x8** point to the exception report record and exception context record, respectively.

The exception report record contains the exception number, and exception address.

The exception context record contains all register values at the time of exception.

The layout for both these records is given in the **BSEXCEPT.H** header file of the OS/2 Programmer's Toolkit.

Most exceptions are generated from a hardware detected exception such as a trap. These are readily intercepted by using the Kernel Debugger **VSF** command. Exceptions may also be generated by the DosRaiseException API. Whatever the source all exceptions will eventually result in a call to **\_xcptrR3ExceptionDispatcher**. This makes this label an excellent break-point for intercepting and filtering any exception that will drive a user's exception handler.

The following example illustrates the use of this break-point, where the system generates a **C0000005** exception following a Trap E in an application program.

```

>> Break on entry to the Ring 3 Exception Handler Dispatcher
##bp _xcptr3exceptiondispatcher

>> Intercept all fatal exceptions
##vsf *
##g
Symbols linked (trape)
Trap 14 (0EH) - Page Fault 0004, Not Present, Read Access, User Mode
eax=00000000 ebx=00000000 ecx=0002059c edx=000a0000 esi=00000000 edi=00000000
eip=0001011c esp=00022e6c ebp=00022e74 iopl=2 rf -- -- nv up ei pl nz ac pe nc
cs=005b  ss=0053  ds=0053  es=0053  fs=150b  gs=0000   cr2=00000000  cr3=001d0000
005b:0001011c 8b00          mov     eax,dword ptr [eax]    ds:00000000=invalid

>> A fatal exception has been intercepted at %1011c
>> Now GT and see the exception dispatcher called.

##gt
eax=00022d18 ebx=00000000 ecx=0002059c edx=000a0000 esi=00000000 edi=00000000
eip=1ff9c8d8 esp=00022bf0 ebp=00022d04 iopl=2 -- -- -- nv up ei pl zr na pe nc
cs=005b  ss=0053  ds=0053  es=0053  fs=150b  gs=0000   cr2=00000000  cr3=001d0000
doscall1:FLAT32:_xcptrR3ExceptionDispatcher:
005b:1ff9c8d8 55          push    ebp                  ;br0

>> %ESP+4 points to the exception report record
>> %ESP+8 points to the exception context record

##dd %esp

```



```
%00022bf0 1ff9c7e9 00022d18 00022d3c 00000000
%00022c00 00000000 2c1a0002 154b0000 00100000
%00022c10 00010002 00000000 032b0000 ffa72212
%00022c20 0058ffaf 2c520066 154b0000 033e0002
%00022c30 52110000 ff9f3130 00000000 00172c52
%00022c40 e91f0000 e9270116 ffa70066 3029ffa7
%00022c50 0008ffa7 e9170000 00570000 00000019
%00022c60 00008000 00000000 00f80000 80000000
```

```
>> The exception report record contains the exception code at
>> offset +0x0 (in this case C0000005).
>> At offset +0xc is the address at which the exception occurred.
>> This agrees with the address seen after VSF intercepted the fatal
>> exception.
```

```
##dd %00022d18
%00022d18 c0000005 00000000 00000000 0001011c
%00022d28 00000002 00000001 00000000 7bb4fc94
%00022d38 ffd9f264 00000007 0000699c fff5416b
%00022d48 00000433 ffd9f264 7b9afe0c ffd9f6378
%00022d58 00000433 000069bc fff54ef2 ffd9f264
%00022d68 fff1f5a50 fe86106c 00000000 fed022d0
%00022d78 00000000 00006a04 fff6d8d9 00000053
%00022d88 00000000 7b9afe3c fff1f5a50 7cf8014c
```

```
>> The context record contains the registers at time of exception.
>> Note the cs:eip at +0xa0 and +0x9c. Also the ss:esp at +0xbc abd
>> +0xb8 and ebp at +0x98.
```

```
##dd %00022d3c
%00022d3c 00000007 0000699c fff5416b 00000433
%00022d4c ffd9f264 7b9afe0c ffd9f6378 00000433
%00022d5c 000069bc fff54ef2 ffd9f264 fff1f5a50
%00022d6c fe86106c 00000000 fed022d0 00000000
%00022d7c 00006a04 fff6d8d9 00000053 00000000
%00022d8c 7b9afe3c fff1f5a50 7cf8014c 7cf80088
%00022d9c 00000001 7cf80088 00000001 7bb2fdb2
%00022dac 00000000 0000150b 00000053 00000053
##d
%00022dbc 00000000 00000000 00000000 00000000
%00022dcc 0002059c 000a0000 00022e74 0001011c
%00022ddc 0000005b 00012216 00022e6c 00000053
%00022dec 00060210 00000000 00000000 000205fc
%00022dfc 000205fc 00020a40 00000000 00000000
%00022e0c 00000000 00000000 00000000 000a0000
%00022e1c 00002000 00000000 00090000 00022e50
%00022e2c 00011fa8 000205fc 00090000 00000000
```

#### Dos32Exit and DosR3ExitAddr

Both these labels provide good breakpoints to catch an application terminating normally.

Dos32Exit is the entry point for the DosExit API.

DosR3ExitAddr is the entry point in DOSCALL1.DLL, called when an application issues the **return** statement to return to the system.

#### Win32SetErrorInfo

This API is called by PM whenever it needs to record a PM error. When this is used as a break-point, the double-word at **%esp+0x4** contains the PM Error code about to be recorded.

#### NWDHandler

This symbol is the entry point to the trap 2 interrupt handler. The [IDT](#) entry for trap 2 contains a Task Gate that points to NWDHandler. When NWDHandler receives control the Task Register will contain the selector for the current [TSS](#). The link field of the current TSS will contain the previous value of the TR, where the processor saved the current registers when the interrupt occurred.

Frequently NMI interrupts are associated with disabled code and obscure hardware or software problems. It can be useful on these occasions to set up a KDB.INI file with the following commands to display information when the trap 2 occurs. This is particularly advantageous when dealing with NMI interrupts caused by the NMI Watch Dog timer firing.

```
bp nwdhandler,"? 'curr tss';dt tr:0;? 'prev tss';dt #(wo(tr:0)):0"
```

**Note:**

When the first NMI occurs, the following would be displayed:

```
curr tss

eax=00000000 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000 edi=00000000
eip=fff4074c esp=00000400 ebp=00000000 iopl=0 -- -- -- nv up di pl nz na po nc
cs=0170  ss=1ea0  ds=0168  es=0168  fs=0000  gs=0000   cr3=001dd000
ss0=0000  esp0=00000000  ssl=0000  esp1=00000000  ss2=0000  esp2=00000000
ldtr=0000  link=0010  tflags=0000  i/o map=ffff
ports trapped: 0-ffff
prev tss

eax=000002ff ebx=139b0000 ecx=00000400 edx=00009ae8 esi=139b993c edi=139d0400
eip=1b7228fe esp=0006eaaa ebp=0006eee0 iopl=2 -- -- -- nv up ei pl nz na po nc
cs=005a  ss=004a  ds=0053  es=0053  fs=150b  gs=0000   cr3=001dd000
ss0=0030  esp0=00006d80  ssl=0000  esp1=00000000  ss2=0036  esp2=0000f000
ldtr=0028  link=0000  tflags=0000  i/o map=dfff
ports trapped: 0-ffff
##
```

The register values when the NMI occurred are displayed under the label *prev tss*.

After NDWHandler has processed the NMI it performs a task-switch back to the previous TSS, but only after editing the previous TSS to ensure that control is passed to TRAPCommonFaultEntry. The task switch is effected using IRETD with the NT flag set in EFLAGS. This leaves the NMI TSS' EIP pointing at the instruction following the IRETD at approximately NWDHandler+25. To allow more than one NMI to be handled the instruction following the IRETD is a JMP NDWHandler. Therefore whenever the NMI TSS' EIP doesn't point to the NDWHandler entry point it is a sure indication that at least one NMI has occurred.

## Trap and Exception Processing

The fine detail of exception management by OS/2 is complex. However the principles are easy to grasp. This section gives an overview of OS/2 Exception Management sufficient to provide the reader with a technique for intercepting exceptions in user code under the Kernel Debugger.

### Exception Definition

Exceptions may be summarised as follows:

- Exceptions refer either to:
  - Hardware Traps and Faults - INTEL defined.
  - Software generated exceptions - OS/2 and User defined.
- Each Hardware Exception has an associated vector, which the processor uses to index the **IDT** to give control to the appropriate system exception handler.
- OS/2 Converts Traps and Faults to software exceptions. For example, traps 0xd and 0xe are converted to exception 0xc0000005.
- Software exceptions are generated from three sources:
  1. Converted Hardware Traps and Faults.
  2. Software Signals.
  3. Software Exceptions from DosRaiseException.
- Exceptions occur for both normal and abnormal reasons. In the normal case additional processing is required to be executed in a manner transparent to the main line code. Examples of this are:

Page fault exceptions.

Trap 1 and 3 for system trace

387 Co-processor emulation

VDM privileged instruction emulation

In the abnormal case, an error condition has been detected. If the error cannot be corrected then either a process or the system dies depending on whether the error can be isolated to a particular process. Usually traps and faults in ring 0 code result in system termination. Bad parameters passed in system APIs may cause the kernel to trap. The system recovers by directing an exception 0xc0000005 to a process. Unless the process can handle this exception, it dies.

- Full details of OS/2 defined exceptions are given in [OS/2 System Exception Codes](#).

### Exception Logic

The essential logic for exception handling is as follows:

- If the processor generates a hardware exception then control is given to the first level exception handler pointed to by the [IDT](#) descriptor that corresponds to the hardware exception vector.
- If the [Kernel Debugger Vector Commands](#) have been specified without the fatal flag then first level exception handlers have been replaced by the Kernel Debugger routines. These may give control to the debugging console or enter the normal system handlers if interception criteria are not satisfied.
- The non-debugger first level routines perform any specific processing for the current exception, for example processing single step and breakpoint traps.
- If full recovery is possible then the first level routines exit with an IRET instruction.
- In most cases control passes from the first level trap handlers to **TrapCommonFaultEntry**. This performs common processing for all hardware exceptions. If recovery is possible, for example by satisfying a page fault or making a segment present, then this is done and control returned to the interrupted code.

If recovery is not directly possible or further special processing is required then control passes to one of the following second level exception handlers:

V8086 Emulation for instruction emulation.

VDM Exception Handler to reflect non-fatal exceptions back to the VDM using its IDT.

Process Fatal Fault Handler (**\_TRAPProcessFatalFault**) for non-kernel mode code (**InDos=0**).

Kernel Fault Handler for kernel code (**InDos=1**)

Special handlers for Co-processor handling, NMIs etc..

- The Kernel Fault Handler checks for the presence of a local fault handler by inspecting **TSDpfnFault**. If this is non-zero then passes control to the local fault handler, otherwise it passes to the System Fatal Fault handler (**SystemFatalFault**).
- This System Fatal Fault handler will enter the Kernel Debugger (if in a non-RETAIL kernel), otherwise it will call and Device Drivers that have registered for notification of fatal system faults, then exit to the panic routine with a formatted message - usually the IPE trap screen. Once in panic the system will not dispatch any more threads. If TRAPDUMP or REIPL are specified then these are acted on otherwise the system waits to be re-booted.
- The Process Fatal Fault handler will check for fatal fault interception by the Kernel Debugger (**VSF command**) and enter the kernel debugger if interception criteria are satisfied. Otherwise user exception processing begins, if it is not possible to dispatch user exception handlers then **DelayHardErr** is called immediately to build the trap screen and wake the Hard Error process. Normally control passes to the **\_XCPTBuildR3DispatcherStack**.
- **\_XCPTBuildR3DispatcherStack** is entry-point for all kernel initiated exceptions to be sent to user. It is responsible for massaging the users stack so that when the kernel exits, control returns to the Exception Dispatcher (**Dos32R3ExceptionDispatcher** in DOSCALL1.DLL).

The parameters to **XCPTBuildR3DispatcherStack** are:

Trap number or 0x0000ffff for S/W generated exceptions  
Exception number  
Count of exception info parameters  
Pointer to the array of exception info parameters  
Boolean, if true then exception is non-continueable.  
Pointer to any additional nested report record

The parameters to **Dos32R3ExceptionDispatcher** are:

Trap number  
Pointer to the [exception report record](#)  
Pointer to the [exception context record](#)

If no exception registration records exist for the current thread then the thread enters termination and the Exception Dispatcher is not called.

- The Exception Dispatcher runs the chain of exception registration records, anchored from the **TIB** of the current thread. Each registered user exception handler is called in turn (via an intermediate routine, **\_xcptExecuteUserExceptionHandler**). The return code (excetion disposition) passed back by the exception handler is examined. If it specifies **XCPT\_CONTINUE\_EXECUTION** then control returns to the kernel via **Dos32ExceptionCallBack**, whereupon the thread's stack is prepared for returning to the interrupted program. If **XCPT\_CONTINUE\_SEARCH** is specified then the next exception handler in the chain is dispatched. When the last exception handler has been dispatched (and all have returned **XCPT\_CONTINUE\_SEARCH**) then control passes to the kernel via **Dos32ExceptionCallBack** and the thread is terminated. **XCPT\_CONTINUE\_STOP** may be returned by a debugger via **DosDebug** to indicate that debugger handler the exception and that exception handler scheduling should be halted immediately.

The values for the various return codes are as follows:

Name	Bit Mask	Description
XCPT_CONTINUE_SEARCH	0x00000000	Exception not handled
XCPT_CONTINUE_EXECUTION	0xFFFFFFFF	Exception handled
XCPT_CONTINUE_STOP	0x00716668	Exception handled by debugger (via DosDebug)

The following additional return codes are used by internal expection handlers to manage nested exceptions.

Name	Bit Mask	Description
NESTED	0xf0f0f0f0	An exception occurred while an exception was active
COLLIDED_UNWIND	0x0f0f0f0f	Indicates collided unwinds
EXIT_UNWIND	0x65796C4B	Indicates the end of an exit unwind

- Dos32ExceptionCallBack** is the kernel entry-point that is called after all user exception handlers have been called. It is passed the following parameters:

Trap number  
Pointer to the [exception report record](#)  
Pointer to the [exception context record](#)  
Exception disposition

This entry-point calls **\_xcptExceptionCallBack** which either takes the default action according to the exception disposition and the exception type (fatal or non-fatal). The either user's context is retored from the exception context record or control passes to **\_xctDefaultAction**. The latter action usually implies proces termination but also may result in a process or system dump being initiated.

For fatal exceptions **\_xctDefaultAction**. calls **DelayHardError** to format the trap screen information from the context record and wake the Hard Error process. It also writes the **POPUPLOG.OS2** entry and calls the Kernel Debugger if the [VSU command](#) has been specified.

- Local Fault Handlers are exception handlers registered by kernel routines. Typically one is registered on entry to the kernel by an API call, and de-registered on exit. If a Local Fault Handler cannot resolve the fault then it will call panic if a serious system fault has occurred, or **\_XCPTBuildR3DispatcherStack** if user code is at fault, for example when a bad parameter supplied to an API by an application program causes the kernel to trap.
- Multiple nested Local Exception Handlers may be registered. When the system calls a Local Exception Handler, the current handler deregisters itself and reinstates the previous nested handler. This is done by restoring the previous handler address from the top-most [long-jump buffer](#) saved in **TSDpljmp** and updating **TSDpfnFault**. The register values saved in the long-jump buffer are restore then then fault handler returns to the main-line code that registered. **EAX** is set by the fault handler to indicate that an

error occurred. When no more local exception handlers are registered then **TSDpfnFault** is zeroed, thereby de-registering all local exception handlers. Local Exception Handlers are always deregistered on exiting the kernel.

- The **DosRaiseException** API is called to create a user exception. This passes control to **\_xcptExceptionCallBack** and normal user exception processing follows.

These details are summarised in the following diagrams:

[Exception Registration Records](#)

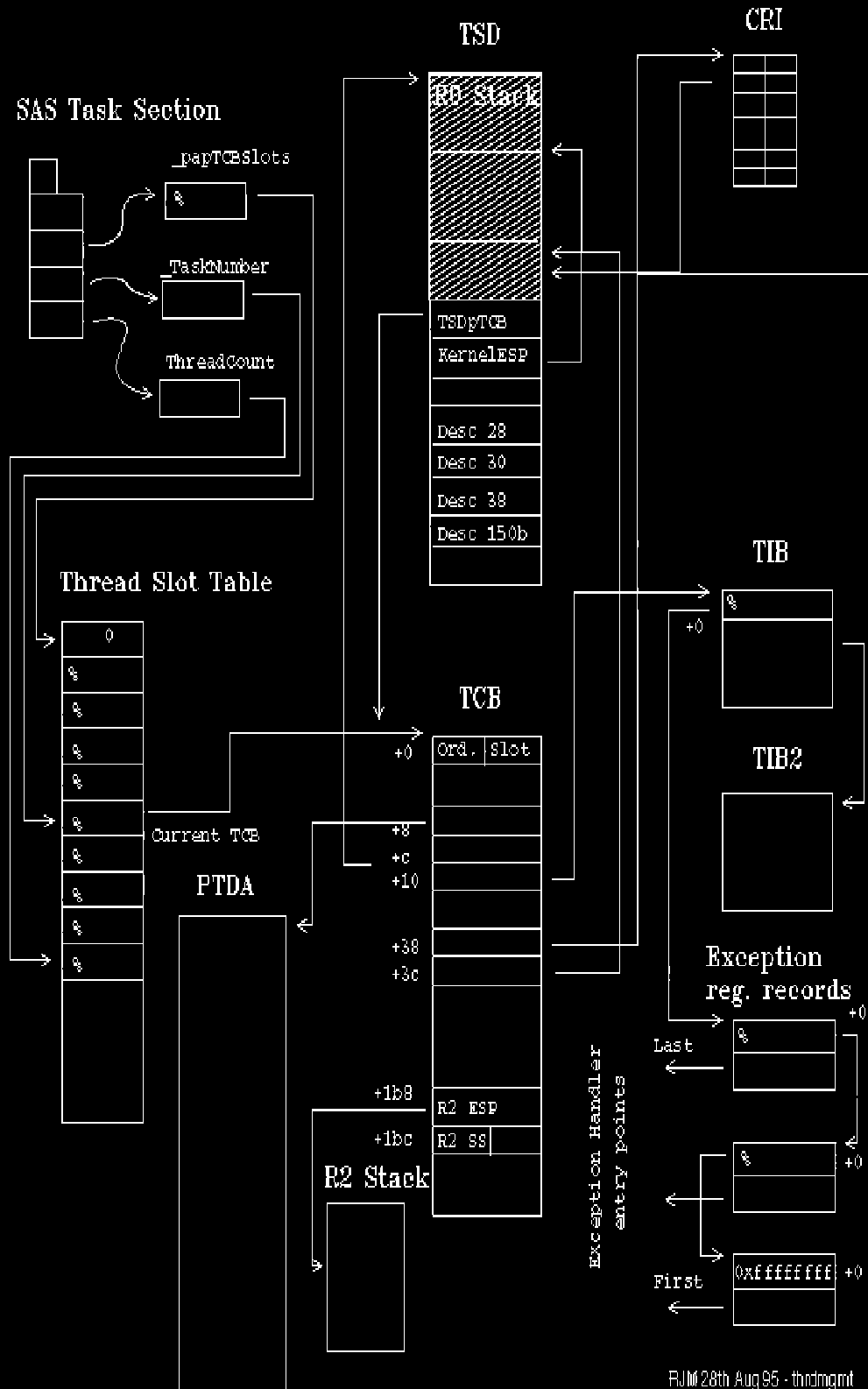
[OS/2 Exception Exception Management - Overview](#)

[Exception Handler Stack Frames](#)

-----

## Exception Registration Records

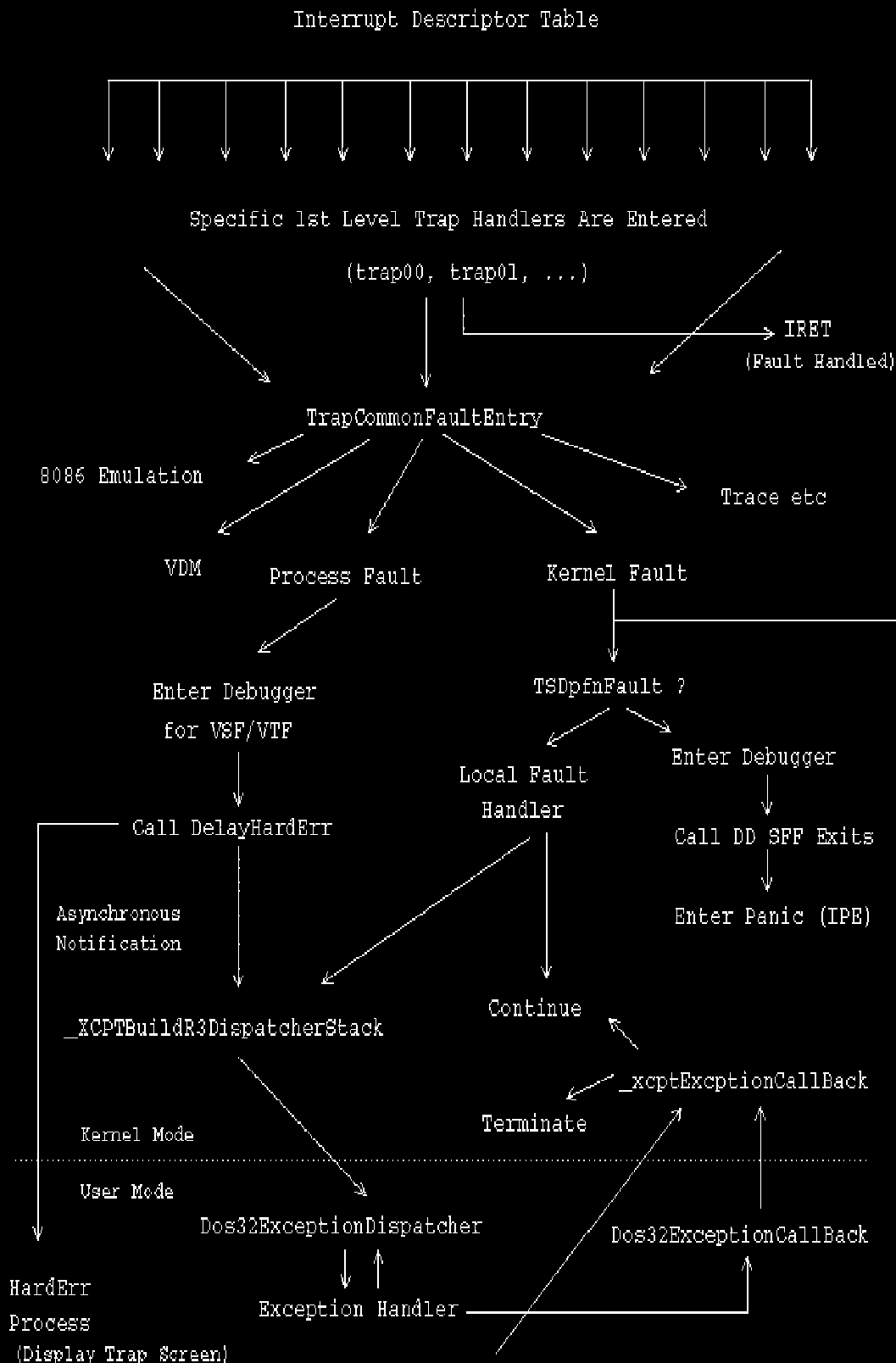
# Thread management



---

## OS/2 Exception Exception Management - Overview

## Exception Handling - Overview

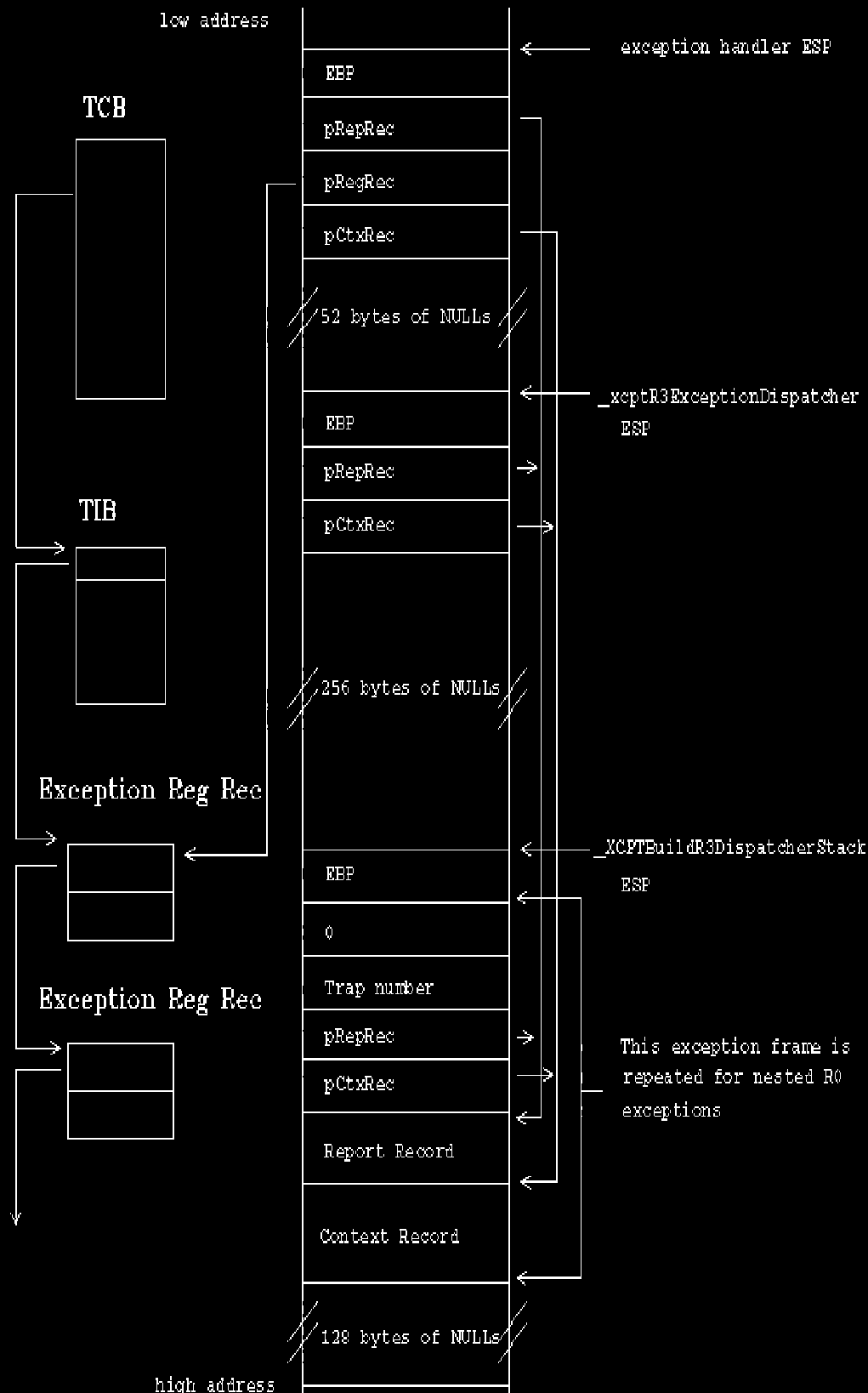




---

# Exception Handler Stack Frames

# Exception Handler Stack Frames



---

# Intercepting Exceptions and Traps

The following list provides guidelines for intercepting traps and exceptions under the Kernel Debugger for various circumstances:

Fatal exceptions occurring in application ring 2/3 code.

**BP \_XCPTBuildR3DispatcherStack** will trap every software and hardware exception. The breakpoint is in the kernel, so use **.R** to display the registers at the time of the exception. This break-point works regardless of whether exception handlers are registered.

**Note:** If the exception is generated through use of an API (bad parameter or **DosRaiseException**) then the **CS:EIP** will point after the call gate instruction.

Fatal Hardware Traps and Faults in application ring 2/3 code.

**VSF \*** will intercept all such exceptions at the point of the exception.

Fatal Hardware Traps and Faults in ring 0 code.

**VTF \*** will intercept all such exceptions at the point of the exception, providing no Local Fault Handler has been registered.

All ring 0-3 traps and faults.

**VT \*** will intercept them all.

All application ring 2/3 code traps and faults.

**VS \*** will intercept them all.

Exceptions in application ring 2/3 code that will drive exception handlers.

**BP \_xcptr3ExceptionDispatcher** will be intercepted if any are registered, but this will be called once to process the entire chain.

Each User Exception Handler.

**BP \_xcptExecuteUserExceptionHandler** will be called to dispatch each exception handler. Alternatively use the registration records from the TIB to locate the entry point of a given exception handler.

Post User Exception Handling

**VSU** will intercept and unrecovered fatal exception delivered to ring 2 and 3 after exception handler processing.

**Note:**

User exception handlers can be disabled under the Kernel Debugger by locating the TIB, then storing 0xffffffff at offset 0x0, which is the pointer to the exception registration record chain. The chain is terminated by 0xffffffff and can be re-worked manually for debugging purposes - provided that the system is not already processing an exception for this thread.

---

# Dump Formatter User Guide

The Dump Formatter is an interactive line-mode utility that supports a variety of commands for extracting and displaying information from a system dump. There are two versions of the Dump Formatter:

**df\_ret.exe**                      The Dump Formatter for dumps from systems running either the **RETAIL** or **HSTRICT** kernels.

**df\_deb.exe**                      The Dump Formatter for dumps from systems running the **ALLSTRICT** kernel.

**Note:** Refer to the [Kernel Debugger User Guide](#) for a discussion on the different OS/2 Kernels.

Each of the two Dump Formatters is generated for each build of the OS/2 kernel. Thus the Dump Formatter is system level and fix-pack level dependent, in a similar way to the debug kernels. Several base versions of the Dump Formatters are distributed with the **OS2PDP** package. Versions of the Dump Formatter that apply to a particular fix-pack may be obtained from the following sources:

- The OS/2 Base Product CDROM for WARP is distributed with the **ALLSTRICT** kernel and Dump Formatter. (For the initial release of WARP this was only available on the US version of WARP).
- The Developer Connection CDROM - this may be ordered through the Developer Assistance Program (DAP) or the System Library Subscription Service (SLSS).
- From your local IBM Marketing Representative.
- Calling the SDM BBS at USA 407-443-8000.
- For IBM internal users by FTP to the SDM at node:

```
sdm.bocaratton.ibm.com
or
9.83.12.237
```

Logon using Id and Password **Anonymous**. A list of files is contained in **files.bbs**.

The Dump Formatter has a named pipe interface that allow it to be controlled from another program. This is exploited by the PMDF program, which is also distributed with the OS2PDP package.

PMDF provides:

- A PM interface to the Dump Formatter.
- Automatic Dump Formatter version management.
- The ability to log output to a file.
- Use of Drag and Drop on Dump Formatter output to the PMDF commands line.
- A [REXX interface](#) that allows REXX EXECs to issue Dump Formatter commands and capture their output.
- [Process Dump](#) Formatting.

The command set supported by Dump Formatter is very similar to that of the Kernel Debugger. In many cases they share common commands. These are documented in the [Kernel Debugger and Dump Formatter Command Reference](#).

## Taking a System Dump

A system dump may created by any of the following means:

- Manually by using the command sequence:  
`Ctrl-Alt-F10-F10` or `Ctrl-Alt-Numlock-Numlock`
- Automatically when an application or system trap occurs. See the description of the [TRAPDUMP](#) CONFIG.SYS setting for details.
- Directly from an application program by using the [DosForceSystemDump](#) API.
- Directly from a physical device driver by calling the **VectorSDF** address returned by the **DevHlp\_GetDOSVar** helper service.
- Under the kernel debugger from the debug console by giving control to the **RASRST** routine. See [Forcing a System Dump From the Kernel Debugger](#) for further details.

### Note:

Whilst it can be advantageous to take a dump when the system trace is active, it is not a pre-requisite. Occasionally system trace has been found to caused other traps.

## Dump Formatter Installation

The Dump Formatter may be installed together with PMDF by using the installation procedure supplied with the OS2PDP package. Alternatively copy the \*.EXE files to either a private directory or a directory in your current PATH. The only files the Dump Formatter accesses implicitly are [Symbol Files](#), which if used, are convenient to have installed in the same directory as the \*.EXE program files.

The command line syntax for Dump Formatter is as follows:

```
DF_RET      dumpfile
DF_DEB      -P pipename
```

The parameters have the following meaning:

***dumpfile***

The file name of the (decompressed) dump to be analysed. If a path is not prefixed to the file name then the Dump Formatter assumes the current path. See [Dump Decompression](#) below.

***-P pipename***

The name of a named pipe through which Dump Formatter output and commands are channelled.

**Note:** This parameter is intended for use when **df\_ret.exe** or **df\_deb.exe** is started from another program using the DosExecPgm API.

**Note:**

If no parameters are entered then the Dump Formatter give a syntax message. This message implies that a COM port may also be used as an interface, but this has not been implemented.

When the Dump Formatter is started it displays the build level of the system from which the dump was taken and then the build level of the formatter. If these do not match unpredictable results may occur. However, if the levels are close then it is probably safe to use the Dump Formatter, though not guaranteed.

If the incorrect type of Dump Formatter is used, for example retail Dump Formatter with a **ALLSTRICT** dump, then the Dump Formatter will probably trap. If it does not, then an error message will appear.

In general the dump formatter traps for one of three reasons:

- The reasons stated above, where there are type and level mismatches.
- The dump file is incomplete or corrupted.
- The Dump Formatter stack overflows.

The latter problem usually occurs when the [.P command](#) is used. This is sometimes circumvented by using the **EXEDHR** utility to increase the stack size of the Dump Formatter. Another approach is to use the **%PS** REXX to display each thread slot individually.

As part of the initialisation sequence, the Dump Formatter attempts to load [symbol files](#), from the current directory, for each module that was loaded on the dumped system.

**Notes:**

Windows, WINOS2 and DOS symbol files are not usable under the Dump Formatter. However, the **SYMLST** REXX exec in the tools directory of the accompanying CD-ROM may be used to list a symbol file. This can sometimes be used in conjunction with the Dump Formatter provided that at least one location of a module or its data can be determined absolutely.

Symbol files not present in the current directory may be manually loaded using the [WA command](#). The syntax and function of this command differs subtly from the Kernel Debugger equivalent:

- Under Dump Formatter names are symbol file names unlike Kernel Debugger where they are symbol map names. This allows relative path names to be used.
- Under Dump Formatter **WA** reads the symbol file, whereas under Kernel Debugger it is just marked active provided it was loaded when the module was loaded.

The Dump Formatter prompts for command input with a single # sign. Unlike the Kernel Debugger this is not used to signify the processor mode or whether paging is enabled. Consequently the Dump Formatter always assumes that the current processor mode is Protect Mode

with Paging Enabled. The user must therefore explicitly prefix **segment:offset** addresses in Virtual 8089 mode with an ampersand (&).  
Commands may be interrupted by pressing the ESC key.

## Dump Decompression

Dumps may be taken either to a dedicated FAT hard disk partition or to diskette. For details on setting up the dump partition refer to the [TRAPDUMP CONFIG.SYS](#) command description.

Dumps taken to a hard disk partition may be used directly by Dump Formatter or PMDF.

Dumps taken to diskette have their data compressed and have to be *decompressed* to produce a single dump file. This may be done from within PMDF by selecting the **New** option of the **File** pull-down. PMDF offers the additional facility of decompressing diskette dumps directly from diskette images created by OS2IMAGE. See [PMDF File Menu](#) below, for details.

Sometimes PMDF fails to decompress a dump. This normally occurs when diskette 1 has not been re-inserted to complete the dump process. If this happend the PATCHDMP utility may be used to correct the dump header on the fist diskette. PATCHDMP may be found on the accompanying CDROM.

If PMDF is not being used then the **DCOMP** command may be used to decompress a dump. The syntax for **DCOMP** is as follows:

NDCOMP	/f	source drive	file name	.
<i>It</i>	Undocumented.			
<i>source drive</i>	Specifies the drive where the <b>DUMPDATA.nnn</b> file will be found. This may specify either a hard disk drive or a diskette drive. The <b>DUMPDATA.nnn</b> files from a diskette dump may be copied to a hard drive root directory before using <b>NDCOMP</b> .			
<i>file name</i>	The target dump file name including path information.			

## PMDF Installation

PMDF provides a convenient front-end to the Dump Formatter. By installing **PMDF** in an appropriate directory structure it is able to select automatically the correct version of Dump Formatter for the dump to be analysed.

PMDF is installed by using the installation of the **OS2PDP** package. The resulting directory structure is as follows:

\PMDF\	PMDF.EXE	PMDF.INF	PMDF.HLP	PMDFMSG.DLL	PMDFVERS.LST	*.CMD
\GA21\	DF_RET.EXE	DF_DEB.EXE	*.SYM	*.SDF		
\GA21MR1\	DF_RET.EXE	DF_DEB.EXE	*.SYM	*.SDF		
\Warp\	DF_RET.EXE	DF_DEB.EXE	*.SYM	*.SDF		
\Warp_fp\	DF_RET.EXE	DF_DEB.EXE	*.SYM	*.SDF		
.						
.						
.						
.						
.						

Each version of OS/2 is represented by a subdirectory containing the Dump Formatters and symbol files and structure definition files for that version.

The home directory contains the PMDF executables and help files, the version control file - PMDFVERS.LST, and any REXX EXECs to be installed in their default directory.

### **PMDFVERS.LST**

More versions of the Dump Formatter may be installed by creating a new subdirectory for the new Dump Formatter and adding an entry to the PMDFVERS.LST file. Each entry of this file corresponds to an OS/2 build level or version. The format of an entry is as follows:

```
relative path:build level:descriptive text
```

### **Notes:**

The path is relative to the home directory.

The build level is the internal system build level and may be determined either by browsing the OS2KRNL load module and searching for the text **@#IBM:n.nnn#@** near the end of the module, or by using the BLDLEVEL utility in the OS2 directory. The **VER /R** command is not reliable since it only reports the base version level, not the fix-pack version level, in some releases.

The directory structure in the example above would be represented by the following entries:

```
ga21:6.514;OS/2 2.1 General Availability
ga21mr1:6.617;OS/2 2.11 MR1
warp:8.162;Warp
warp_fp:8.200;Warp Full Pack
```

-----

## PMDF Menus and Options

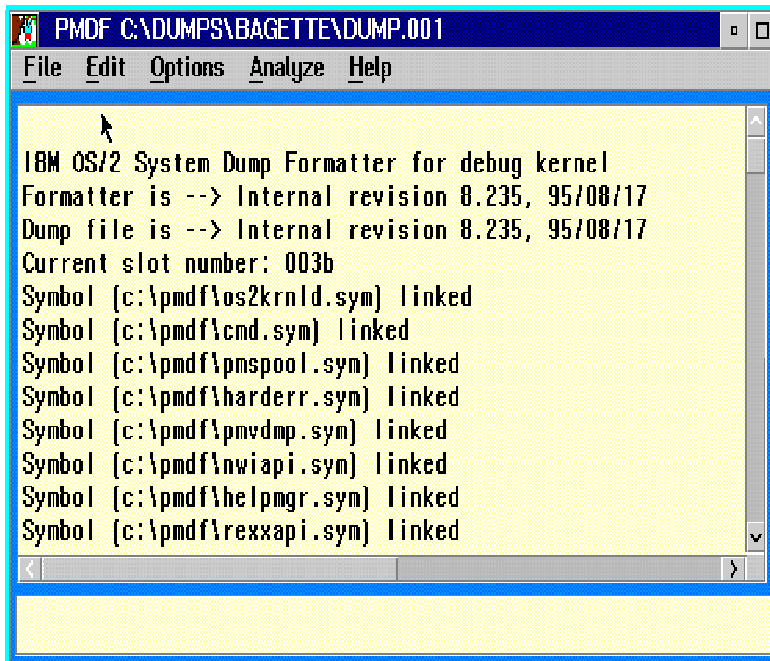
PMDF offers a number of facilities from its pull-down menus and also from the mouse buttons.

From the Keyboard **Ctrl-C** and **Esc** serve to interrupt the Dump Formatter.

### **Warning:**

Do not use the Dump Formatter [Q command](#). Under PMDF this may cause PMDF to hang. To terminate the Dump Formatter either quit PMDF from the system menu or select another dump for processing.

The PMDF screen appears as follows:



---

## PMDf File Menu

The File pull-down menu offers the following options:

### New Dump

Select this option to decompress a new dump.

#### Notes:

For diskette dumps the **DUMPDATAnnn** files may be copied for a directory on the hard drive and decompressed from there.

PMDf has the ability to decompress diskette images created by **OS2IMAGE** without re-creating the original diskettes. To use this facility each of the image file must be named **image.nnn** where nnn is a numeric sequence number that corresponds to the disk number.

### Open Dump

This option prompts the user for the dump file name then invokes Dump Formatter.

### Log Output

This option prompts the user to start or stop logging output to a file. Data may be appended to an existing log file.

### Save Output

This option allows the user to save all output displayed in the PMDF scrollable window.

### Connect

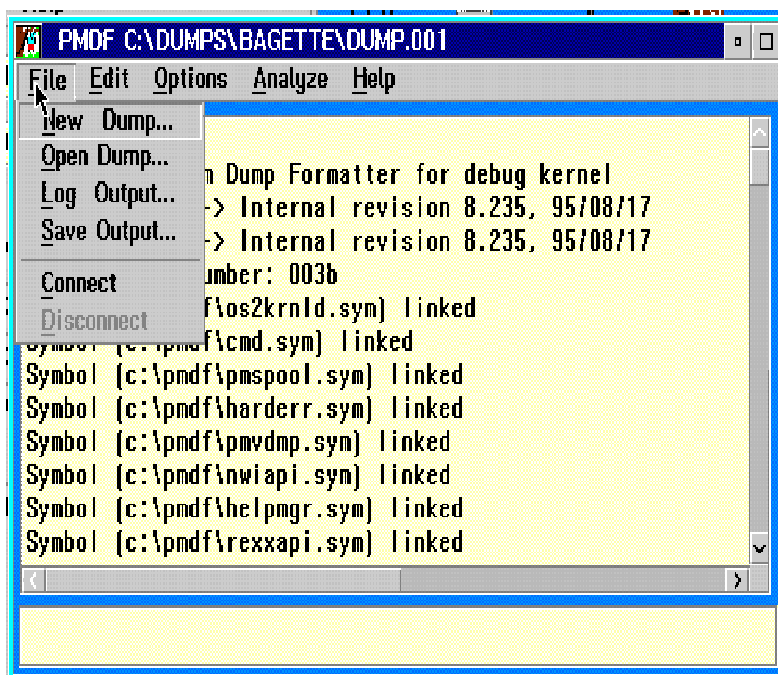
Connect allows PMDF to be used as a terminal emulator to drive a Kernel Debugger session. See the [Kernel Debugger User Guide](#) for more information.

### Disconnect

Disconnect terminated the communications session with the Kernel Debugger.

The following diagram illustrates the File pull-down menu options.



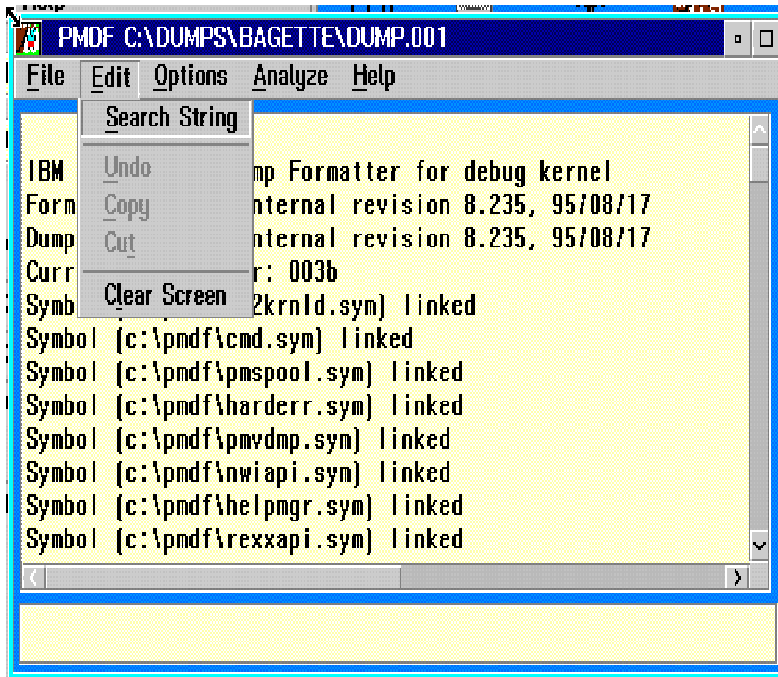


## PMDf Edit Menu

The Edit pull-down menu offers the following options:

Search String	Locates text within the scrollable window.
Undo	Reverse the previous Edit Cut action.
Copy	Copy marked text to the clip board.
Cut	Move marked text to the clip board.
Clear Screen	Clears the scrollable window of all text. This is not a reversible action.

The following diagram illustrates the Edit pull-down menu options.



## PMDf Options Menu

The Options pull-down menu offers the following options:

### Font Settings

This allows font selection for displayed output.

### Function Keys

This provides a menu to predefine function keys as strings of Dump Formatter command strings. Commands may be separated by a semi-colon.

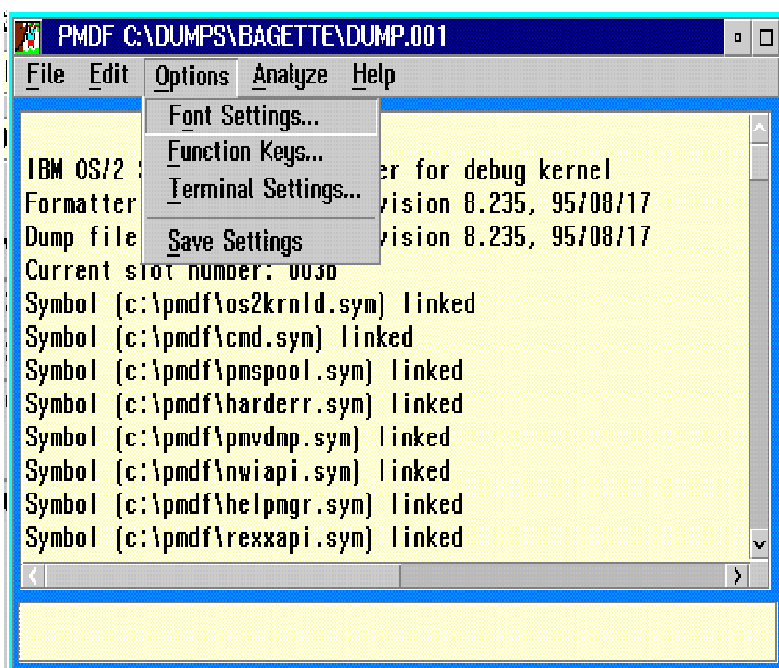
### Terminal Settings

Allows the communications parameters to be specified for when the Connect option of the File pull-down is selected.

### Save Settings

This will save the current options in PMDF.INI for use next time PMDF is started.

The following diagram illustrates the Options pull-down menu options.



## PMDf Analyze Menu

The Analyze pull-down menu offers four selections, each of which displays its own menu selection. Where parameters are required they should be highlighted by double-clicking mouse button 1 on text in the scrollable window.

### CAUTION:

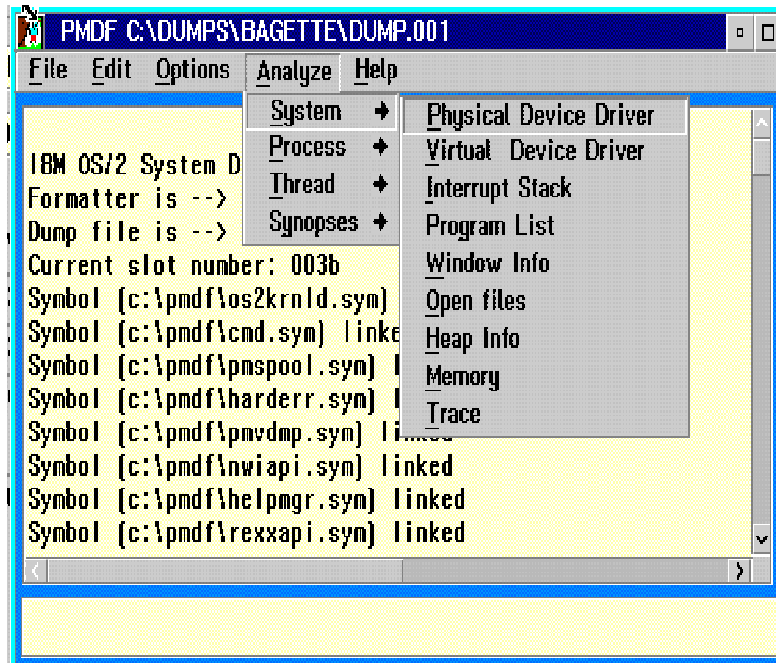
The output from the Analyze options needs to be interpreted with care. Some options are precise since they follow control block chains anchored from the [SAS](#), for example the Physical Device Driver Chain and Kernel Heap. Others depend, for correct results, on correct symbols being loaded. Some options, for example those that display stacks, are more speculative in what they display.

Before these facilities are relied on, the user should thoroughly acquaint themselves with the manual techniques that belie their function. This information is available in the course materials that comprise the first section of this Handbook.

The following selections are available:

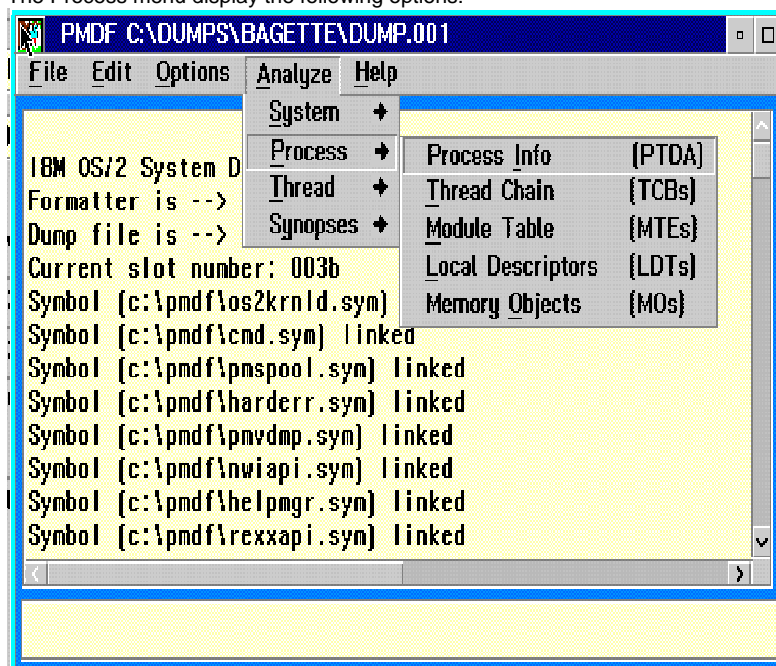
System

The System menu displays the following options:



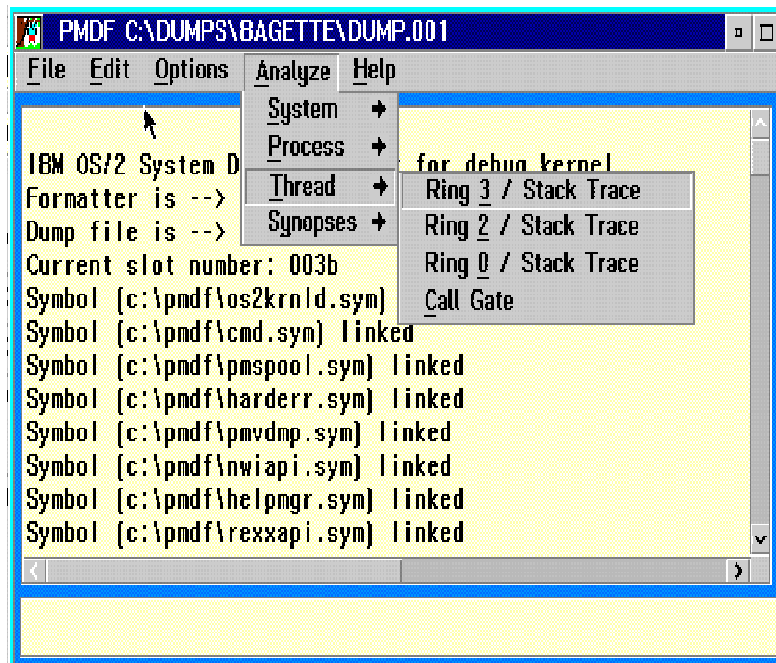
## Process

The Process menu display the following options:



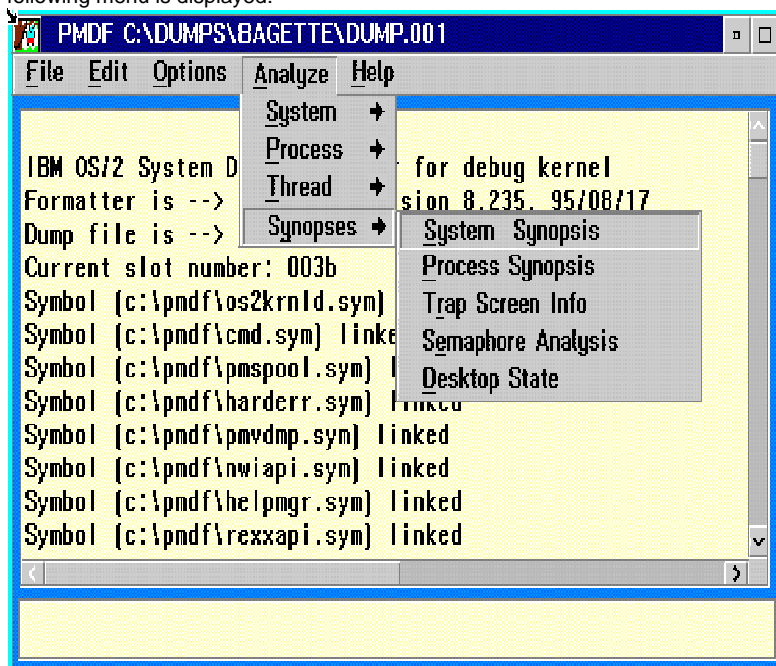
## Threads

The Threads menus dumps stacks related to a given thread. The following menu is displayed:



## Synopsis

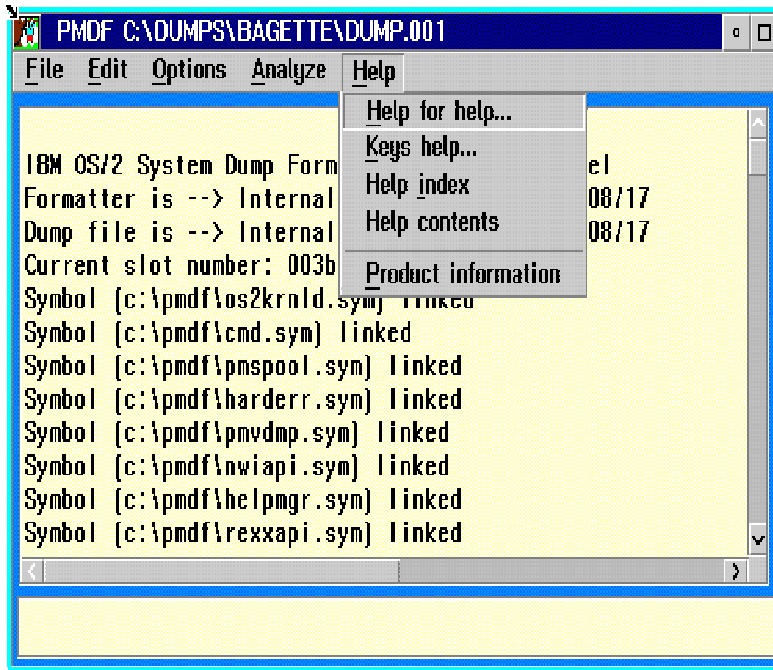
This offers a miscellaneous collection of options, the most important of which is the Trap Screen display. The following menu is displayed:



## PMDf Help Menu

The Help pull-down menu offers standard help facilities.

The following diagram illustrates the Options pull-down menu options.



---

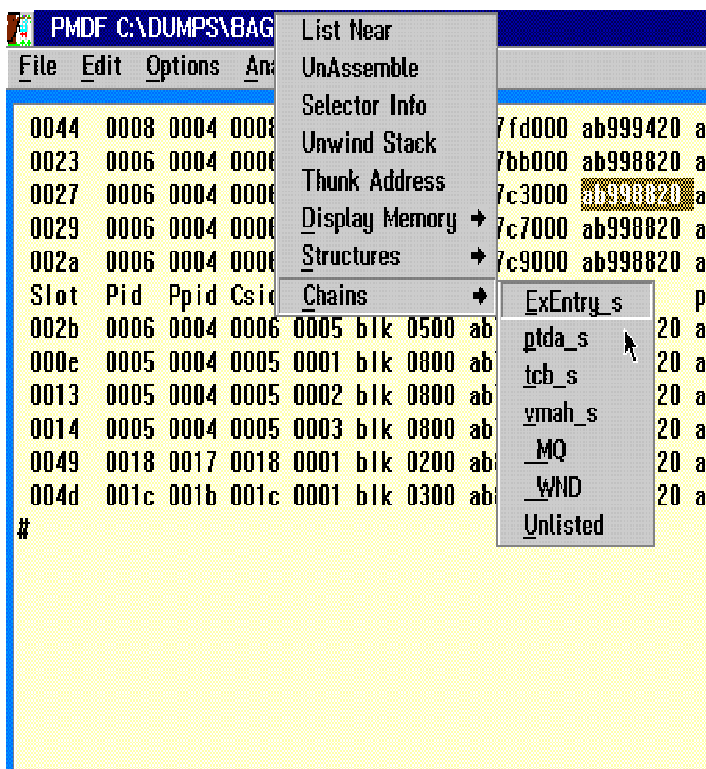
## PMDf Mouse Options

Standard CUA mouse selection and highlighting are implemented. Marked items may be dragged and dropped onto the command line.

A double-click with mouse button 1 will highlight a blank delimited string.

A single click with mouse button 2 will display pop-up menu whose items take the highlighted text in the scrollable output window as input.

The following diagram shows an example of the mouse pop-up menu. In this example the Structures option is displayed. This particular option acts as a supplement to the Dump Formatter [.D command](#). For it to work correctly, the Structure Definition Files (\*.SDF) are required to be present in the same directory as the Dump Formatter. These files are build level dependent and will only display correct information if matched to the dump level. There is no validation performed by these displays. The user must ensure that an appropriate input address is highlighted.



## PMDF REXX interface

PMDF provides a REXX interface that allows REXX EXECs to issue Dump Formatter commands and capture their output in REXX variables. EXECs are able to display output on PMDF's scrollable output window, command line and enter

EXECs are invoked by entering the REXX EXEC name, with optional directory information, prefixed with a '%' character from the PMDF command window. If the exec is not installed in a directory in the PATH or in the same directory as PMDF then it must be prefixed with the fully qualified path name. For example:

```
%SEGTAB 123
```

```
%C:\MYEXECS\TEST1 parm1 parm2
```

It is also possible to use relative path expressions thus:

```
%..\SEGATB 123
```

If a path has to be specified when passing an exec name as a parameter to another exec then quotation marks around the path and file name will be required.

PMDF implements its interface to the Dump Formatter by creating a REXX subcommand environment. The REXX **address** instruction allows an EXEC to execute and capture the output from a Dump Formatter command by addressing this subcommand environment.

The syntax and parameters for this implementation of the **address** instruction are:

```
address df 'CMD' <output> <df_cmd>
```

Where:

<output>

is the name of a stem to a REXX compound variable that will be assigned to capture output from the Dump Formatter command.

"output.0" will be set to the number of lines. "output.n" will contain the nth line of output.

<df\_cmd>

is the dump formatter command and parameters.

Parameters following the EXEC name will be passed to the EXEC as a one parameter string.

A number of general purpose EXECs are provided in the OS2PDP package on the CD-ROM accompanying this book. These are:

**RUNCHAIN**

Generalised Control Chain Running EXEC.

**PS**

Generalised EXEC for executing Dump Formatter commands per thread slot.

**TEMPLATE**

A Template EXEC containing a collection of subroutines useful for writing other EXECs.

There are also a number of example EXECs that format control blocks and illustrate how to use the REXX interface and the subroutines contained in TEMPLATE.

-----

## The RUNCHAIN EXEC

### Syntax

```
RUNCHAIN <addr> link(<offset>,<s>) stopvalue(<stop>) chain(<nnn>) exec(<cmd>)  
          print(<file>)
```

This exec provides a generalised control block chaining facility, where at each hop of the chain a command or exec may be executed. The starting address and link offset are required. Other parameters are optional. The parameters to RUNCHAIN are:

<addr> is an address expression of the start of the chain

<offset> specifies the decimal or hexadecimal offset of the linking address. Default is 0

<s> specifies the length of the linking field as: D (double) or W (word) - Default is D

<stop> specifies a termination value for the linking field. This take precedence over <chain> and may be specified as a hexadecimal or decimal value.

<nnn> specifies the maximum number of chain hops to traverse. Default is 10

<cmd> specified a command to be executed at each hop. If the command is prefixed with a % then an exec is executed. @L will cause the linear address of the current block to be substituted. Default is DD @L L4.

<file> specifies a print file to which the output will be copied.

### **Note:**

Hexadecimal values are specified as 'nn'x

As an example: suppose the linear address of an MTE is %fff2bde0. MTEs are linked at +c in os2 2.1. To run the chain of MTEs displaying 8 double words do the following:

```
%RUNCHAIN %fff2bde0 link(c) exec(DB @L L40)
```

The resulting output would be appear thus:

```
Block 1 at %FFF2BDE0
```



```
%fff2bde0 00060002 fff2bdfc fff2bfc3 fe0aldac
%fff2bdf0 0000b980 00000000 00010000 00000000
```

Block 2 at %fe0aldac

```
%fe0aldac 02600002 fcace908 fe0aldfc fe083e40
%fe0aldbc 4498b1c6 0000000d 0000003a fe0addf0
```

Block 3 at %fe083e40

```
%fe083e40 024e0002 fcac52f0 fe083e70 fe0adef8
%fe083e50 4498b1c6 00000005 00000038 fdf40fac
```

Block 4 at %fe0adef8

```
%fe0adef8 01aa0002 fcac07a0 fe0adf14 fdf61cc8
%fe0adf08 0498b1c8 00000000 00000035 4d495405
```

Block 5 at %fdf61cc8

```
%fdf61cc8 01a80002 fca9ad58 fdf61ce4 fdf61d68
%fdf61cd8 0498b1c8 00000000 00000036 53595307
Chain run successfully for 5 hops
```

To format the first 40 MTEs in the chain do:

```
%RUNCHAIN %fff2bde0 link(c) exec(.lmo @L) chain(40)
```

-----

## The PS EXEC

### Syntax

```
PS <s1> <s2> <cmd> <parms> </cmd> <parms> .....
```

This is the Per-Slot exec. It will repeatedly execute a DF command string or REXX exec for each thread slot in the range specified. The linear addresses of slot related control blocks (TCB, PTDA and TSD) may be specified symbolically in the command string so that the correct address will be substituted for each slot traversed by PS.

The parameters to PS are:

<s1>	Starting (hexadecimal) slot number
<s2>	Ending (hexadecimal) slot number or *, which signifies highest active slot in the system.
<cmd>	is any string of DF commands separated by ; or a single REXX exec prefixed by %.
<parms>	are any valid parameters where @TCB, @PTDA and @TSD are substituted with their corresponding linear addresses. @disp is the scheduler's ESP relative to the TSD. N.B @disp is only defined when page table entries are present for the TSD.

### Example 1:

Display priority information (on a 2.11 system) for slots 30 to 33 where priority class is at TCB+e4, priority delta is at TCB+e5 and dispatching priority is a word at TCB+e8.

Enter:

```
%PS 30 33 DB @TCB+e4 L2; DW @TCB+e8 L1
```

```

Slot 30
Warning: not all addresses are present
DB %7BA8FE78+E4 L2; DW %7BA8FE78+E8 L1

%7ba8ff5c 02 0f      ..
%7ba8ff60 020f

Slot 31
DB %7BA9002C+E4 L2; DW %7BA9002C+E8 L1

%7ba90110 02 00      ..
%7ba90114 0200

Slot 32
Warning: not all addresses are present
DB %7BA9002C+E4 L2; DW %7BA9002C+E8 L1

%7ba90110 02 00      ..
%7ba90114 0200

Slot 33
DB %7BA90394+E4 L2; DW %7BA90394+E8 L1

%7ba90478 03 00      ..
%7ba9047c 0800
ps ended rc: 0

```

**Note:**

For slot 30 a warning message is issued because in this instance .s30 gave an error because slot 30 page tables were swapped out.

-----

## The TEMPLATE EXEC

Template is not intended to be executed. Rather, it is a model for creating new execs. It contains a number of generally useful subroutines used in other execs.

Currently included in TEMPLATE are the following subroutines:

```

linaddr <address>
    Converts an address expression to a linear address (without the % prefix). If storage cannot be referenced then a null
    string is returned.

getstor <h>,<a>,<s>,<f>
    Retrieve a byte, word or double word from storage. If storage can't be retrieved then the DF error msg is returned.

    <h>                is a dump handle
    <a>                is a DF address expression
    <s>                is the size specified as: B, W or D
    <f>                is the optional output format, which may be specified as C for character, N for
                        decimal or X for hexadecimal string. X is the default.

gethstr <h>,<a>,<l>
    Retrieve a string of hex bytes from storage. If storage can't be retrieved then a null string is returned. The string is
    returned as a concatenated string of bytes.

    <h>                is a dump handle
    <a>                is a DF address expression
    <l>                is the length of storage to retrieve

```

getbytes <h>,<a>,<l>

Retrieve a one or more bytes from storage. If storage can't be retrieved then a null string is returned. The string is returned as a string of bytes separated by blanks.

<h>	is a dump handle
<a>	is a DF address expression
<l>	is the length of storage to retrieve

getwords <h>,<a>,<l>

Retrieve a one or more words from storage. If storage can't be retrieved then a null string is returned.

<h>	is a dump handle
<a>	is a DF address expression
<l>	is the length of storage to retrieve

getdwords <h>,<a>,<l>

Retrieve a one or more double words from storage. If storage can't be retrieved then a null string is returned.

<h>	is a dump handle
<a>	is a DF address expression
<l>	is the length of storage to retrieve

getqwords <h>,<a>,<l>

Retrieve a one or more quadruple words from storage. If storage can't be retrieved then a null string is returned.

<h>	is a dump handle
<a>	is a DF address expression
<l>	is the length of storage to retrieve

format <name>,<offset>,<base>,<type>,<desc>

Format a field from a control block and returns the value of the field.

<name>	is the field name.
<offset>	is a relative hex offset (prefix with + or -)
<base>	is the base address of the control block
<type>	is the field type (b=byte, w=word, d=double word)
<desc>	is a description of the field.

fmblock <name>,<offset>,<base>,<type>,<number>,<desc>

Formats a table of bytes, words or double words imbedded in a control block.

<name>	is the field name.
<offset>	is a relative hex offset (prefix with + or -)
<base>	is the base address of the control block
<type>	is the field type (b=byte, w=word, d=double word)
<number>	is the number of entries in the table
<desc>	is a description of the field.

-----

## Process Dump Formatter

PMDf provides a Process Dump Formatter facility which is invoked automatically when the Open option of the File pull-down menu is selected against a Process Dump.

The Process Dump Formatter offers a limited subset of the full Dump Formatter command set. These are:

<b>.D</b>	Display storage in Bytes, Words or Double-Words.
<b>.DL</b>	Display LDT entries.
<b>L</b>	List, Symbols, Maps and Symbol Groups
<b>.LM and .LMO</b>	Display Module Table Entries and Object Tables
<b>.MA</b>	Display Arena Records for storage dumped.
<b>.MO</b>	Display Object Records for storage dumped.
<b>.ML</b>	Display Information on Dumped Memory.

**Note:**

This command does not perform the same function as the similarly named Kernel Debugger **.ML** command, which formats VM Alias Records.

<b>.P</b>	Display threads.
-----------	------------------

**Note:**

Unlike the Dump Formatter and Kernel Debugger version of this command, **.P** is used to select the thread ordinal within the dumped process. Thus for single thread processes **.P 1** is the only valid combination.

<b>.PB</b>	Display thread Block IDs.
------------	---------------------------

**Note:**

Unlike the Dump Formatter and Kernel Debugger version of this command, **.PB** is used to select the thread ordinal within the dumped process. Thus for single thread processes **.PB 1** is the only valid combination.

<b>R</b>	Display registers for each thread.
----------	------------------------------------

<b>.S</b>	Set default thread slot.
-----------	--------------------------

**Note:**

Unlike the Dump Formatter and Kernel Debugger version of this command, **.S** is used to select the thread ordinal within the dumped process. Thus for single thread processes **.S 1** is the only valid combination.

<b>W</b>	Load and Unload Symbol files
----------	------------------------------

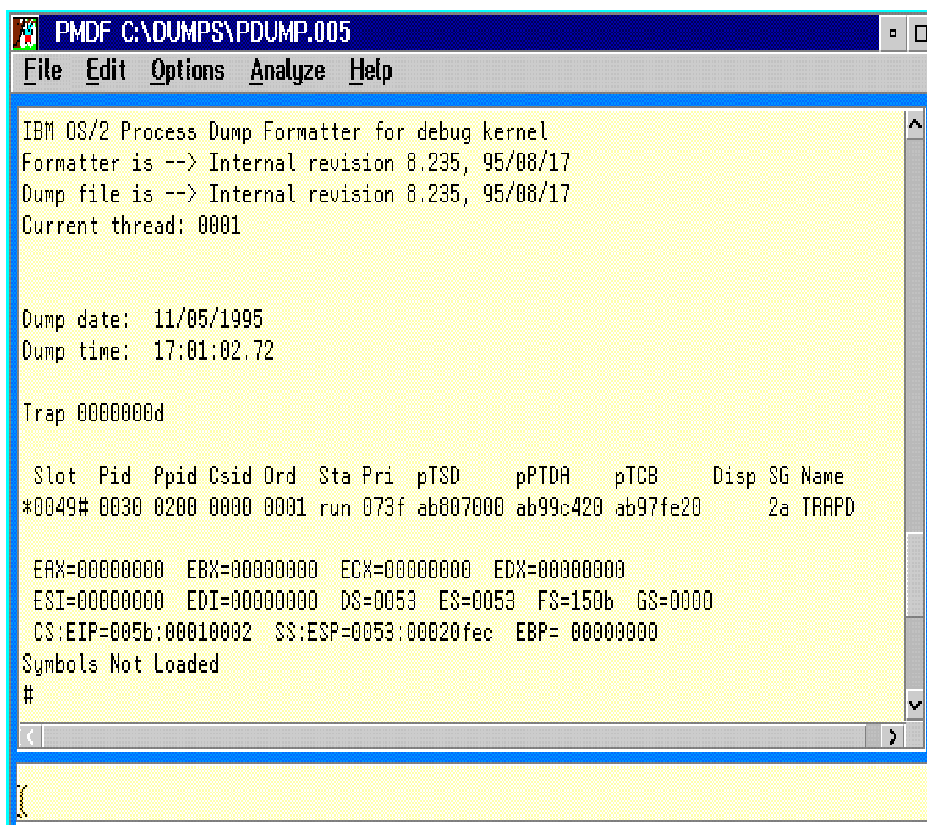
<b>?</b>	Syntax help for internal commands
----------	-----------------------------------

<b>.?</b>	Syntax help for external (dot) commands.
-----------	--

**Note:**

Except where noted above, the command set for the Process Dump Formatter does not support any of the optional parameters supported by their equivalent Kernel Debugger commands.

When a Process Dump is loaded PMDF displays the following screen:



**Note:**

The data and time of the dump are displayed.

If the dump was created because of a trap then the trap number is displayed otherwise the trap number is shown as **ffffff**.

The current thread slot and register are shown last.

The Analyze pull-down menu differs from the standard PMDF Analyze facility. This offers the following choices:

**Registers**

This performs the **R** command for each thread dumped.

**Task Summary**

This performs a **.P** command followed by an **R** command for each thread dumped.

**Local Descriptors**

This performs a **DL** command.

**Virtual Memory Control Blocks**

This performs a **.MA** and **.MO** command.

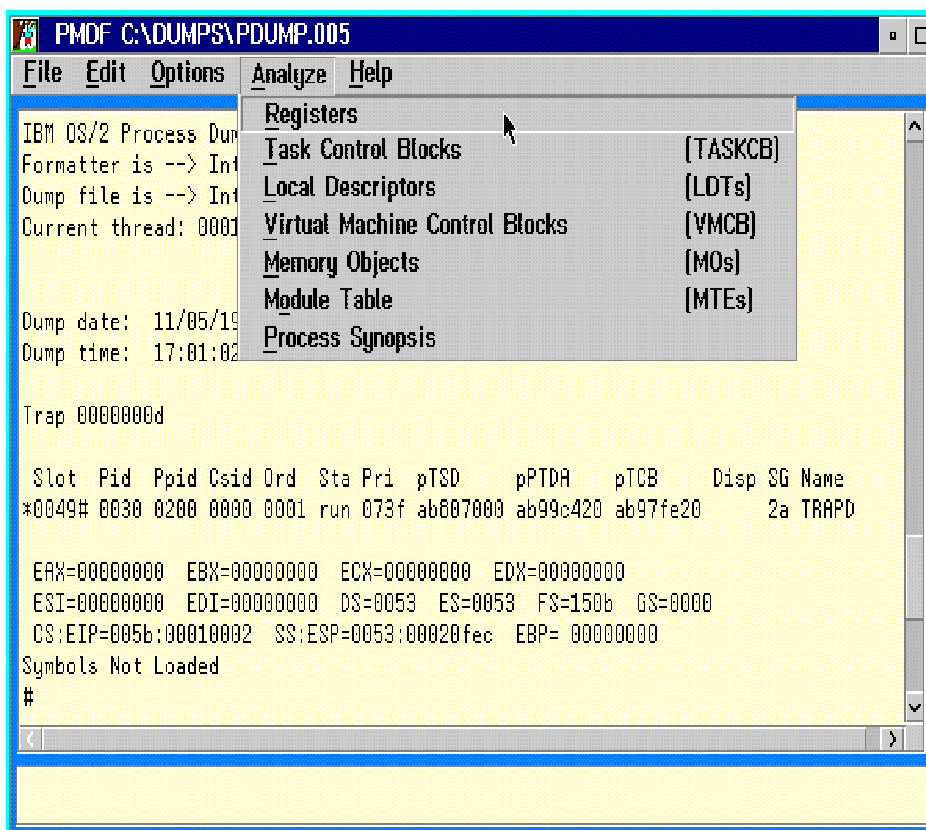
**Module Table**

This is a much more extensive version of the **.LMO** command. The entire MTE and SMTE for each module dumped is formatted.

**Process Synopsis**

This formats the entire Process Dump, including dumping all memory in Byte format.

The Analyze option menu appears as follows:



For information on taking and controlling Process Dumps see:

The CONFIG.SYS [DUMPPROCESS](#) command.

The [DosProcessDump](#) API.

## Kernel Debugger and Dump Formatter Command Reference

The Kernel Debugger and Dump Formatter share a common subset of commands which comprises the vast majority of the combined command set. The following symbols will be used to denote to which tool commands are applicable:



Dump Formatter



Kernel Debugger

References to some system control offsets blocks are made in the descriptions of the commands. For the sake of brevity, the **ALLSTRICT** version of the OS/2 WARP 3.0 kernel is assumed and in some cases the equivalent **RETAIL** and **HSTRICT** kernel offsets are given in parentheses. For example:

**JFN\_pTable** (PTDA +0x5b8 (H/R: +0x5b0))

The reader should refer to [System Reference](#) for control block layouts of the **ALLSTRICT**, **HSTRICT** and **RETAIL** kernels for OS/2 WARP 3.0 and OS/2 2.11 kernels.

Commands are categorised into two classes:

[Internal](#)

Internal commands begin with an alphabetic character. They are control program independent in the sense that they

relate only to the Intel 80x86 hardware architecture.

#### External

External commands are prefixed with a period. They relate to the software environment under analysis and are dependent on the data structures of the operating system environment.

For a description of the conventions used in the syntax diagrams, see [Syntax Diagrams - Notation](#)

Complex expressions may be used where substituted values are required. The rules governing expressions are described in [The expression evaluator](#).

-----

## Syntax Diagrams - Notation

The command syntax descriptions for the Dump Formatter and Kernel Debugger use a graphical notation, which is now in common use. The diagrams should be read as a *road map* starting at the sign:

and ending at the sign:

,

The command verb and options are shown in upper case type.

Parameter values to be supplied by the user are shown in lower case. The rules governing the use of complex expressions are described under [The expression evaluator](#).

Continuation of the syntax diagram is shown by:

at the break, and

at the beginning of the continuation line.

Expansion of syntax into detailed subsections is indicated thus:

section name

Expanded section begins:

section name:

Expanded section ends:

See below for a more detailed description and example of subsections.

Mutually exclusive options where a **non-mandatory** selection is required are shown thus:

A  
B  
C

Here at most one of A, B or C may be selected.

Mutually exclusive options where a **mandatory** choice is required are shown thus:

A  
B  
C

Multiple selections are shown:

A  
B  
C

D  
E  
F

One or more of A, B or C is optional, whereas at least one of D, E or F is required.

If a separator is required between parameters then it is shown in the diagram. For example, a comma is required between each selection in the following:

,

A  
B  
C

,

D  
E  
F

For the Dump Formatter and Kernel Debugger spaces between parameters options are optional.

Ordered non-exclusive selection lists and parameters are shown in the order they must be specified.

X

,

A  
B  
C

,

D  
E  
F

H  
I

= value

Here, X must be specified first, followed optionally by zero or more of A, B or C separated by commas, followed by at least one of D, E or F separated by commas, followed optionally by H or I and finally by the character = and a quantity substituted for **value**.

The following examples would be correct interpretations of this last syntax diagram:

X D = 55  
XA,B I=444

Where complex diagrams require splitting into multiple sections, the sections are identified by a lower case *italic* name. For example:

*section 1*

*section 2*

*section 1:*

A

B  
C

In this example the syntax for *section 1* is exclusive with *section 2*. The options for **section 1** are shown at the label ***section 1:***



---

# The Expression Evaluator

The Kernel Debugger and Dump Formatter expression evaluator supports a variety of arithmetic, boolean and addressing operators to form a value to be substituted into a command parameter may be derived. The atomic entities used within expressions may be string or numeric in type. Arithmetic expressions may be used with addressing separators to represent a physical, linear, selector:offset or segment:offset address's. Certain conventional values may be represented in expressions by [mnemonics](#).

[Symbols](#) defined by symbols files may also be used to represent either their equivalent address operator and address arithmetic value combination or constant arithmetic value in command line expressions.

---

## String Expressions

These are identified by being enclosed in either single or double quotes. A string may contain any keyboard character including quotation marks, which must be duplicated so as not to act as a string terminator. Examples are:

```
'this is a sting'
'That''s an other example'
"and so is this"
```

Where there is no ambiguity then terminating quote may be omitted.

---

## Arithmetic Expressions

The expression evaluator will accept numeric values in a decimal, hexadecimal, binary and octal notation. These are indicated thus:

<b>nnnnnnY</b>	Binary number <b>nnnnnn</b> .
<b>nnnnnnO</b>	Octal number <b>nnnnnn</b> .
<b>nnnnnnQ</b>	Alternative notation for octal number <b>nnnnnn</b> ,
<b>nnnnnnT</b>	Decimal number <b>nnnnnn</b> .
<b>nnnnnnH</b>	Hexadecimal number <b>nnnnnn</b> .

The base suffix may be in upper or lower case.

The default base when a suffix is omitted is hexadecimal.

The following represent the same number, expressed in each of the permissible forms:

```
31
31t
1fh
37o
37q
10001111y
```

Arithmetic expressions are of three types:

<b>Absolute</b>	<p>An arithmetic expression that resolves to a numeric value.</p> <p>Absolute expressions may be formed from numeric values using arithmetic <a href="#">binary</a> and <a href="#">unary</a> operators and <a href="#">in-built functions</a> together with parentheses (), to influence evaluation order.</p>
<b>Boolean</b>	<p>Boolean expressions are ones that resolve to either a <b>TRUE</b> or <b>FALSE</b> value.</p> <p>Boolean expression may be formed from arithmetic expressions using boolean <a href="#">binary</a> and <a href="#">unary</a> operators together with parentheses (), to influence evaluation order.</p> <p>Boolean expressions may be used as absolute values in arithmetic expressions. Whereupon <b>TRUE</b> assumes the value 1 and <b>FALSE</b> 0.</p>
<b>Address</b>	<p>An arithmetic expression that resolves to one or two numeric values that represent a linear, physical, segment:offset or selector:offset address.</p> <p>Address expressions are formed from absolute expressions using <a href="#">addressing separators</a>.</p>

**Note:**

The expression evaluator allows arithmetic values to be expressed in hexadecimal. A potential conflict may occur where symbol names exist that begin with letters: **a - f**. For example, a linear address expressed as **%fe1234** may be rejected with the message:

```
Invalid expression
```

where a symbol **f** or **fe** is defined. To avoid this conflict prefix the hexadecimal numeric value with a zero, thus:

```
%0fe1234
```

If the same error message persists then the address refers to either paged out or unallocated virtual memory.

## Binary Operators

### Arithmetic operators:

The following binary operators are permissible in any arithmetic expression:

<b>*</b>	Multiplication
<b>/</b>	Integer division
<b>MOD</b>	Modulo or remainder operator
<b>+</b>	Addition
<b>-</b>	Subtraction
<b>AND</b>	Bitwise AND
<b>XOR</b>	Bitwise exclusive OR
<b>OR</b>	Bitwise OR

### Boolean operators:

The following binary operators are permissible in any boolean expression:

>	Greater than
<	Less than
>=	Greater than or equals
==	Logical equality
!=	Logical inequality
&&	Logical AND
	Logical OR

---

## Unary Operators

### Arithmetic operators:

The following unary operators are permissible in any arithmetic expression:

NOT	Bitwise ones complement
—	Bitwise Twos complement

### Boolean operators:

The following unary operator is permissible in any boolean expression:

!	Logical negation
---	------------------

---

## In-built Functions

The following in-built functions operate in a single address expression operand:

SEG	Returns the segment or selector portion of an address that resolves to either a <b>&amp;segment:offset</b> or <b>#selector:offset</b> form.
OFF	Returns the offset of an address the resolves to either a <b>&amp;segment:offset</b> or <b>#selector:offset</b> form.
BY	Returns one byte from an address location.
WO	Returns one word from an address location.
DW	

Returns one double word from an address location.

#### POI

Returns one double word far pointer (selector:offset or segment:offset address) from an address location. The low order word returned is treated as the offset. The high order word returned is treated as a selector or segment based depending on the default addressing mode. See the [D command](#) for more information.

#### PORT

Returns one byte from an 8-bit I/O port address.

#### WPORT

Returns one word from a 16-bit I/O port address.

#### Example:

```
DD %(dw(%7abcde0+10))
```

Display the storage whose linear address is at location %7abcdf0.

-----

## Address Separators and Address Expressions

The following separators may be used with absolute expressions to form elements of an address:

**&**

Segment prefix

**#**

Selector prefix

**%**

Linear address prefix

**%%**

Physical address prefix

**:**

A segment/offset address separator.

**|**

[Thread slot number](#) qualifier.

Where virtual addresses map to different physical addresses in different processes (typically private arena data and shared arena instance data) then | may be used to qualify the address by thread slot number.

#### **Note:**

This qualifier is ignored by the Dump Formatter

#### Examples:

```
%ebp
```

The value of **EBP** assumed to be a linear address.

```
%%10034
```

The physical address at location 10034.

```
38 | #1f:0
```

The selector:offset address 1f:0 in the context of slot 38.

-----

# Evaluation Order

Expressions are evaluated left to right by applying the following order of precedence to operators, separators and in-built functions:

1.        ( )
  2.        | :
  3.        & # % %% \_ ! NOT SEG OFF BY WO DW POI PORT WPORT
  4.        \* / MOD
  5.        + -
  6.        > < >= <=
  7.        == !=
  8.        AND XOR OR
  9.        && ||
- 

## Mnemonics and Symbols

Symbols defined in symbol files may be used in any arithmetic expression. Absolute symbols (that is, symbols of constants) are treated as absolute expressions. Other symbols are treated as address expressions. Symbols are activated using the [WA command](#).

The in-built register mnemonics supported by the Kernel Debugger and Dump Formatter are:

- 16-bit registers:  
ax, bx, cx, dx, si, di, bp, ip, pc
- 32-bit registers:  
eax, ebx, ecx, edx, esi, edi, ebp, eip
- Segment registers:  
cs, ds, es, fs, gs, ss
- Flag registers:  
flg, eflg
- Control registers:  
cr0, cr2, cr3
- GDTR register:  
gdtb, gdtl
- IDTR register:  
idtb, idtl
- Task control registers:  
tr, ldtr, msr
- Debug registers:

dr0, dr1, dr2, dr3, dr4, dr5, dr6

- Test registers:

tr6, tr7

These may be used as absolute expressions for the current register value. See the [R command](#) for information on displaying and setting current register values and for the definition of the register mnemonics.

The Kernel Debugger also defines mnemonics:

- br0, br1, br2, ..., br9

to represent the addresses of breakpoints defined by the [BP](#) and [BR](#) commands.

The expression evaluator allows the prefix [@](#) to a symbol name to distinguish it from a similarly named mnemonic name. For example, [@ax](#) refers to the symbol [ax](#), whereas [ax](#) refers to the [ax](#) register value.

Similar conflicts may also arise between hexadecimal values and symbols. These may be avoided by prefixing and hexadecimal numeric value with a zero.

-----

## Internal Commands

The following comprise the set of internal commands:

<a href="#">?</a>	Display internal command help
<a href="#">B</a>	Breakpoint command family.
<a href="#">BC</a>	Clear breakpoint
<a href="#">BD</a>	Disable breakpoint
<a href="#">BE</a>	Enable breakpoint
<a href="#">BL</a>	List breakpoints
<a href="#">BP</a>	Set or change a breakpoint
<a href="#">BR</a>	Set a debug register breakpoint
<a href="#">BS</a>	Show time-stamped breakpoint trace
<a href="#">BT</a>	Set time-stamped breakpoint trace
<a href="#">C</a>	Compare memory
<a href="#">D</a>	Dump memory data (default)
<a href="#">DA</a>	Dump memory ASCII data
<a href="#">DB</a>	Dump memory byte data
<a href="#">DD</a>	Dump memory double-word data
<a href="#">DG</a>	Dump global descriptor table
<a href="#">DI</a>	Dump interrupt descriptor table
<a href="#">DL</a>	Dump local descriptor table
<a href="#">DP</a>	Dump Page Tables
<a href="#">DT</a>	Dump Task State Segment
<a href="#">DW</a>	Dump memory word data

<a href="#">DX</a>	Dump 80286 Loadall buffer
<a href="#">E</a>	Enter memory data
<a href="#">F</a>	Fill memory
<a href="#">G</a>	Go
<a href="#">H</a>	Perform hexadecimal arithmetic
<a href="#">I</a>	Input from 16-bit I/O port
<a href="#">J</a>	Execute commands conditionally.
<a href="#">K</a>	Display current stack
<a href="#">L</a>	List maps, groups and symbols
<a href="#">M</a>	Move memory data
<a href="#">O</a>	Output to 16-bit I/O port
<a href="#">P</a>	Process Trace
<a href="#">Q</a>	Quit the Dump Formatter
<a href="#">R</a>	Display/Alter registers
<a href="#">S</a>	Search
<a href="#">T</a>	Trace
<a href="#">U</a>	Unassemble
<a href="#">V</a>	Trap Vectors Command family
<a href="#">W</a>	Add/remove symbol map
<a href="#">Y</a>	Set Kernel Debugger options
<a href="#">Z</a>	Set/list/execute the default command

-----

## ? - Show Internal Command Help or Evaluate an Expression



Display help for internal Kernel Debugger and Dump Formatter commands and evaluate complex expressions.

### Syntax:

```
? [expr string]
```

### Parameters:

*(default)*

Displays a help summary for most of the Dump Formatter and Kernel Debugger internal commands.

### **Note:**

Some of the information displayed is out-of-date.

Two pages of information are displayed with an intervening **--More--** prompt.

***expr***

An expression that resolves to either a simple numeric value or an address using any of the [expression evaluation operators](#). [Symbols of addresses](#) and [symbols of absolute values](#) may be specified.

***string***

A string enclosed in single or double quotes.

**Results & Notes:**

If an expression is specified then it is evaluated. If it resolves to an address then it is displayed in equivalent forms, as follows:

```
sel:offset %linaddr %%physaddr
```

Where:

***sel:offset***

Specifies the selector and offset form of the address if the expression resolves to a ***sel:offset*** form.

***%linaddr***

Specifies the linear address equivalent of the expression if it resolves to either a ***sel:offset*** or ***%linaddr*** form.

***%physaddr***

Specifies the physical address equivalent of the expression. If the expression resolves to a virtual address then the page tables must be present to perform the address translation.

See the [DP command](#) for information on displaying page table entries and the [.l command](#) for information on paging in memory.

if the evaluated expression resolves to an absolute value then it is displayed in hexadecimal, decimal, octal, binary, character and boolean forms. For example:

```
-----
##? 5
05H 5T 5Q 00000101Y '.' TRUE
? bmp_segsize
12H 18T 22Q 00010010Y '.' TRUE
-----
```

**Notes**

Each arithmetic value is suffixed with a modifier that indicates the base used:

- H** Signifies hexadecimal
- T** Signifies decimal (Tens)
- Q** Signifies octal (Qctal?)
- Y** Signifies binary (Yes/no?)

In the last example above, **bmp\_segsize** is an [absolute symbol](#) of value **0x0012** defined in map **OS2KRNL**.

If a string expression is displayed then it is echoed back to the console. For example:

```
-----
##? "This is a way of anotating the debug log from this session's analysis"
This is a way of annotating the debug log from this session's analysis
##
-----
```

**Note:**

Evaluation of simple expressions involving two absolute expressions may be done using the [H command](#).



---

# B - Breakpoint Command Family



The breakpoint family of eight commands provide a means of defining and managing *sticky* breakpoints.

**Syntax:**

B	C	
	D	options
	E	
	L	
	P	
	R	
	S	
	T	

**Parameters:**

C	Clear breakpoints.  See <a href="#">BC command</a> for <i>options</i> .
D	Disable breakpoints.  See <a href="#">BD command</a> for <i>options</i> .
E	Enable breakpoints.  See <a href="#">BE command</a> for <i>options</i> .
L	List breakpoints.  See <a href="#">BL command</a> for <i>options</i> .
P	Set or change breakpoints.  See <a href="#">BP command</a> for <i>options</i> .
R	Set a debug register breakpoint.  See <a href="#">BR command</a> for <i>options</i> .
S	Set a time-stamped breakpoint trace.  See <a href="#">BS command</a> for <i>options</i> .
T	Display a time-stamped breakpoint trace.  See <a href="#">BT command</a> for <i>options</i> .
<i>options</i>	See the associated command for details.

---

# BC - Clear Breakpoints



Clear 1 or more breakpoints.

**Syntax:**

```
BC [ / n * ]
```

**Parameters:**

- n* Breakpoint number to be cleared.
  - \* may be specified to clear all breakpoints.
- See [Breakpoint commands](#) for information on listing and setting breakpoints.

**Results & Notes:**

The specified breakpoints are cleared. No information is displayed.

-----

# BD - Disable Breakpoints



Disable 1 or more breakpoints.

**Syntax:**

```
BD [ / n * ]
```

**Parameters:**

- n* Breakpoint number to be disabled.
  - \* may be specified to disable all breakpoints.
- See [Breakpoint commands](#) for information on listing and setting breakpoints.

**Results & Notes:**

The specified breakpoints are disabled. No information is displayed.

-----

# BE - Enable Breakpoints



Enable 1 or more breakpoints.

**Syntax:**

```
BE n *
```

**Parameters:**

**n** Breakpoint number to be enabled.  
\* may be specified to enable all breakpoints.  
See [Breakpoint commands](#) for information on listing and setting breakpoints.

**Results & Notes:**

The specified breakpoints are enabled. No information is displayed.

## BL - List Breakpoints



List all breakpoints defined by the [BP](#) and [BR](#) commands.

**Syntax:**

```
BL
```

**Parameters:** none.

**Results & Notes:**

**BL** lists the definitions of all currently defined breakpoints. An example of this follows:

```
bl
0 e 0158:00005874 [DOSOPEN] 5 (5) ".P*;G"
1 d 0158:00007384 [DOSHOLDSIGNAL]
2 e %fff461a4 [_tkSchedNext] 12 (15) ".P*"
3 dT %fff474e4 [_PGSwitchContext]
4 d %1a022298 [DOS32WRITE] 10 (10)
5 e W2 0030:0000ffcc [Ppid]
6 dI E1 0000:00000000 5 (5) "DW TASKNUMBER L1"
7 e I1 00002e7
##
```

Breakpoint definitions are of two forms:

1. The general layout of the **BP** breakpoint definition is:

```
n st addr [symbol] pc (mc) "cmd, cmd, ...."
```

2. The general layout of the **BR** breakpoint definition is:

```
n st tn addr [symbol] pc (mc) "cmd, cmd, ...."
```

Each of the fields has the following meaning:

<i>n</i>	The breakpoint number assigned to the given breakpoint.
<i>st</i>	<p>The status of the breakpoint:</p> <p><b>d</b> Disabled breakpoint. See <a href="#">BD command</a>  <b>e</b> Enabled breakpoint. See <a href="#">BE command</a>.</p> <p>The suffix <b>T</b> signifies that the breakpoint is a time-stamp breakpoint created using the <a href="#">BT command</a>.</p> <p>The suffix <b>I</b> indicates that the address has become invalid.</p>
<i>tn</i>	<p>The register breakpoint type (t) and size (n):</p> <p><b>R</b> Read breakpoint.  <b>W</b> Write breakpoint.  <b>I</b> I/O breakpoint.</p> <p><b>Note:</b> I/O breakpoints are only available to Pentium (and later) processors. The support for I/O breakpoints was introduced into the Kernel Debugger with fix pack 29 for Warp 3.0 and base Warp 4.0.</p>
<i>addr</i>	The address at which the breakpoint is defined.
[ <i>symbol</i> ]	The breakpoint offset to the nearest symbolic address, if it exists. See <a href="#">LN</a> and <a href="#">WA</a> commands for information on listing and loading symbol definitions.
<i>pc</i>	The remaining <i>passcount</i> for this breakpoint. If a <i>passcount</i> is not defined then this value is not displayed. See <a href="#">BP</a> and <a href="#">BR</a> commands for more information on passcounts.
( <i>mc</i> )	The initial passcount defined for this breakpoint. If no passcount was defined then this value is not displayed. See <a href="#">BP</a> and <a href="#">BR</a> commands for more information on <i>passcounts</i> .
" <i>cmd, cmd, ...</i> "	A list of commands to be executed when the breakpoint <i>fires</i> . Each command is separated by commas and the entire string is enclosed in quotes. If no command string is defined then this field is not displayed. See <a href="#">BP</a> , <a href="#">BR</a> and <a href="#">Z</a> commands for more information on breakpoint command lists.

## BP - Set or Alter a Breakpoint



Set or re-specify a software *sticky* [breakpoint](#). by inserting an **INT 3** instruction.

### Syntax:

```
BP      addr      passcount      cmd      cmd      ;
BPn                                           "      "
```

### Parameters:

*n*

Explicitly specifies a breakpoint number to be assigned to this breakpoint. A value from 0 to 9 may be specified. If specified there must be no space between the number and the **BP** command.

The default is to assign the lowest available number. If all 10 breakpoint numbers have been assigned then the following message appears:

Too many breakpoints

*addr*

The [address](#) of the breakpoint.

The Kernel Debugger saves the byte of storage at the location specified by *addr* and inserts an **INT 3** instruction in its place.

### **Notes**

Whenever the Kernel Debugger is entered the storage overlayed by any breakpoints is temporarily restored. When the Kernel Debugger gives control back to the system, enabled breakpoints are re-instated.

If *addr* specifies the address of an existing breakpoint then the existing breakpoint is updated with the new parameters.

Each break-point address is recorded with its associated process context. For shared data this is of no consequence. However for private addresses, especially those in the private arena the *addr* may be qualified by [slot](#) number by using the | operator. This acts as a shorthand to save changing contexts using the [.S command](#) in order to set the breakpoint correctly. For example, suppose the current slot is 8, then:

```
BP 31 | %10032
```

is equivalent to:

```
.S 31  
BP 31 | %10032  
.S 8
```

*passcount*

Specifies the number of times the breakpoint may be passed before the Kernel Debugger is entered. Each time the breakpoint is passed the count is decremented by 1 until 1 is reached. When the breakpoint is encountered with a count of 1 then it will *fire* and the Kernel Debugger will be entered. Thus if **passcount** is **5** then the breakpoint will *fire* on the **5th** encounter.

The default passcount is 1, that is, the breakpoint will *fire* on first encounter.

*cmd*

Specifies a command to be executed when the breakpoint *fires*. More than one command may be specified by using a semi-colon separator and enclosing the entire command list in single or double quotes.

If no command string is specified then the default command string, as specified by the [Z command](#) will be executed.

### Results & Notes:

If the specified address is valid then the breakpoint definition is accepted otherwise one of the following messages is generated:

```
Invalid linear address: %nnnnnnnn  
Invalid selector: selector:offset  
Past end of segment selector:offset
```

If the break-point is successfully defined then the in-built mnemonic **BRn**, where **n** corresponds to the break-point number, takes the value of the break-point address. This may be used in any address expression or any command.

### **Note:**

Since **BP** break-points are implemented by the insertion of **INT 3** instructions, it is possible for such break-points to become discarded if the page of code is discarded and subsequently paged back into memory.

If the [.I command](#) is used to swap in a page of code, then the break-points are automatically restored. (In earlier versions of OS/2 it was necessary to specify the **B** option of [.I](#)).

This complexity may be avoided by setting register break-points with the [BR command](#).

---

## BR - Set or Alter a Debug Register Breakpoint



Set or alter a *sticky* [breakpoint](#) using the debug registers.

### Syntax:

```
BR      E      addr      ~
BRn     Wb      pc      ,  "  cmd
        Rb      ,  "  cmd
        Ib      ;
```

### Parameters:

***n***

Explicitly specifies a breakpoint number to be assigned to this breakpoint. A value from 0 to 9 may be specified but from this range only a total of 4 may specify enabled debug register breakpoints.

If a value ***n*** is specified there must be no space between the number and the **BR** command.

The default is to assign the lowest available number. If all 10 breakpoint numbers have been assigned then the following message appears:

```
Too many breakpoints
```

If all four debug registers are in use then the message:

```
Out of debug registers
```

is displayed.

### Note:

A disabled debug register breakpoint does not commit the use of a debug register. Thus more than 4 debug register breakpoints may be defined, but only a maximum of 4 enabled at any time.

See the [BE](#) and [BD](#) commands for information on enabling and disabling breakpoints.

***E***

Specifies that the breakpoint is to *fire* when an instruction at the breakpoint address is fetched for execution.

This is mutually exclusive with the **W** and **R** parameters.

***Rb***

Specifies that the breakpoint is to *fire* when storage at the breakpoint address, for length ***b*** is referenced. ***b*** may specify 1, 2 or 4 bytes and defaults to 1 byte if left blank.

This is mutually exclusive with the **W**, **I** and **E** parameters.

***Wb***

Specifies that the breakpoint is to *fire* when storage at the breakpoint address, for length ***b*** is stored. ***b*** may specify 1, 2 or 4 bytes and defaults to 1 byte if left blank.

This is mutually exclusive with the **R**, **I** and **E** parameters.

***lb***

Specifies that the breakpoint is to *fire* when data is read or written from an I/O port address specified by ***addr***. The length operand, ***lb*** is used by the processor to mask out the low order bits of the I/O address. Thus:

```
BR I2 %13f
```

places an I/O breakpoint on ports 13c - 13f. ***lb*** may specify 1, 2 or 4 bytes and defaults to 1 byte if left blank.

Note that the IO port address must be specified as a linear address, otherwise the expression evaluator will attempt to use the base address of the current **CS** register to resolve the address parameter.

This is mutually exclusive with the **R**, **W** and **E** parameters.

**Note:** I/O breakpoints are only available to Pentium processors. Support for this was introduced from fix pack 29 of Warp 3.0 and base Warp 4.0, however a bug has prevented them from working correctly until Fix Pack 0 of Warp 4.0 and Fix Pack 39 of Warp 3.0.

***addr***

The [address](#) of the breakpoint.

The Kernel Debugger converts the address to a linear address before setting up the debug registers. If the address is invalid the definition is retained but marked disabled and invalid.

**Note:** Real addresses may not be used with debug register breakpoints.

***passcount***

Specifies the number of times the breakpoint may be passed before the Kernel Debugger is entered. Each time the breakpoint is passed the count is decremented by 1 until 1 is reached. When the breakpoint is encountered with a count of 1 then it will *fire* and the Kernel Debugger will be entered. Thus if **passcount** is **5** then the breakpoint will *fire* on the **5th** encounter.

The default passcount is 1, that is the breakpoint will fire on first encounter.

***cmd***

Specifies a command to be executed when the breakpoint *fires*. More than one command may be specified by using a semi-colon separator and enclosing the entire command list in single or double quotes.

If no command string is specified then the default command string, as specified by the [Z command](#) will be executed.

**Note:** The command list **must** be preceded by a comma, unlike the **BP** command where the comma is optional.

#### **Results & Notes:**

If the specified address is valid then the breakpoint definition is accepted and enabled otherwise it is accepted but disabled and one of the following messages is generated:

```
Invalid selector: selector:offset  
Past end of segment selector:offset
```

If the break-point is successfully defined then the in-built mnemonic **BRn**, where **n** corresponds to the break-point number, takes the value of the break-point address. This may be used in any address expression or any command.

-----

## BS - Show Time-stamped Breakpoint Trace



Show the [time-stamped breakpoint trace](#).

#### **Syntax:**

BS

### Parameters:

None.

### Results & Notes:

The time-stamp trace buffer is formatted in LIFO order. The following is an example of the formatted trace:

```
Number of entries = 4284
BP0 381e6ae1a (hex)
BP4 381e68292 (hex)
BP0 381e658d1 (hex)
BP4 381e40559 (hex)
BP0 381e3da7d (hex)
```

### **Notes**

The number of entries is the total accumulated number of time-stamp trace events regardless of wrapping of the (4096 entry) time-stamp trace buffer.

Each entry show the breakpoint number that corresponds to the time-stamped breakpoint that *fired*, the high resolution time stamp in microseconds and a reminder that this value is in hexadecimal.

For information on defining a time-stamp breakpoint see the [BT command](#).

---

## BT - Set Time-stamped Breakpoint Trace



Set a time-stamped breakpoint trace.

### Syntax:

```
BT      addr
BTn
```

### Parameters:

***n***

Explicitly specifies a breakpoint number to be assigned to this breakpoint. A value from 0 to 9 may be specified. If specified, there must be no space between the number and the **BT** command.

The default is to assign the lowest available number. If all 10 breakpoint numbers have been assigned then the following message appears:

```
Too many breakpoints
```

***addr***

The [address](#) of the breakpoint.

The Kernel Debugger saves the byte of storage at the location specified by ***addr*** and inserts an **INT 3** instruction in its place.

### **Notes**

Whenever the Kernel Debugger is entered the storage overlayed by any breakpoints is temporarily restored. When the Kernel Debugger gives control back to the system, enabled breakpoints are re-instated.

If ***addr*** specifies the address of an existing breakpoint then the existing breakpoint is updated with the new parameters.



**Results & Notes:**

If the address is valid then the breakpoint definition is accepted and enabled. When enabled, the time-stamp breakpoint causes the current high resolution system time to be saved in a time-stamp circular trace buffer whenever the breakpoint address is executed.

The trace buffer will record up to 4K entries before wrapping.

Unlike the **BP** and **BR** commands, **BT** does not return control to user when the breakpoint is encountered.

The time-stamp trace may be displayed using the [BS command](#).

C - Compare Memory



Compare up to 64k bytes of memory at two locations in storage.

**Syntax:**

C                  addr1                  n                  addr2                  \

**Parameters:**

- addr1**  
The address of the beginning of the first location to compare with the second. This address is assumed to be in *# selector:offset* format. If the selector is omitted then the current **DS** selector is assumed.
- n**  
The offset from **addr1** of the last byte to compare (that is, the length of the range less 1).
- addr2**  
The address of the beginning of the second location to compare with the first. An [address expression](#) may be specified. This address is assumed to be in *# selector:offset* format. If the selector is omitted then the current **DS** selector is assumed.

**Results & Notes:**

Storage is compared, if no differences are found then the command prompt is displayed. If either of the addresses is invalid then an error message is displayed.

Where differences are found in the address range then they are displayed in the following way:

001f:00000000 57 4f 001f:00000003  
001f:00000001 50 32 001f:00000004  
001f:00000003 57 4f 001f:00000006

The addresses of the two differing locations are displayed outermost and the bytes at those locations are displayed in columns 2 and 3.

D - Display Memory



Display a range of memory from a given address.

**Syntax:**

D                  Dn  
DA                  addr  
DB                  Ln

DW  
DD

### Parameters:

*(default)*

Display memory using the current display format. When the user breaks into the Kernel Debugger the current format is set to Byte display. If the user subsequently executes a **DW**, **DA** or **DD** command then the current format is set to words, ASCII or double-words, respectively. Byte format default may be restored by using **DB**.

**A**

Force memory to be displayed in ASCII format and set the current display format to ASCII. The display is terminated as soon as the first null byte (**0x00**) is reached or the length specification is reached.

**Note:** The current display address is not updated when in ASCII format.

**B**

Force memory to be displayed in Byte format and set the current display format to Byte.

**W**

Force memory to be displayed in Word format and set the current display format to Word.

**D**

Force memory to be displayed in Double-word format and set the current display format to Double-word.

**addr**

The address of the memory location to display. When the user breaks into the Kernel Debugger this defaults the current **DS** selector, offset **0**. If a display command other than **DA** is executed then the current display address is updated to the last displayed address **+1**.

An [address expression](#) may be specified.

**L/n**

The number of bytes, words or double-words to display, depending upon the current display format. If not specified this defaults to **128** bytes, **64** words and **32** double-words respectively.

### Results & Notes:

Memory is displayed according to the selected display format providing the address is valid. If it is not, but the address represents pageable storage then this may be paged in to memory using the [.I command](#).

The following examples show output in the four different formats.

ASCII format:

```
##da 1f:0
```

```
001f:00000000 WP_OBJHANDLE=177110
```

Byte format:

```
##db 1f:0
```

```
001f:00000000 57 50 5f 4f 42 4a 48 41-4e 44 4c 45 3d 31 37 37 WP_OBJHANDLE=177
001f:00000010 31 31 30 00 55 53 45 52-5f 49 4e 49 3d 43 3a 5c 110.USER_INI=C:\
001f:00000020 4f 53 32 5c 4f 53 32 2e-49 4e 49 00 53 59 53 54 OS2\OS2.INI.SYST
001f:00000030 45 4d 5f 49 4e 49 3d 43-3a 5c 4f 53 32 5c 4f 53 EM_INI=C:\OS2\OS
001f:00000040 32 53 59 53 2e 49 4e 49-00 4f 53 32 5f 53 48 45 2SYS.INI.OS2_SHE
001f:00000050 4c 4c 3d 43 3a 5c 4f 53-32 5c 43 4d 44 2e 45 58 LL=C:\OS2\CMD.EX
001f:00000060 45 00 41 55 54 4f 53 54-41 52 54 3d 54 41 53 4b E.AUTOSTART=TASK
001f:00000070 4c 49 53 54 2c 46 4f 4c-44 45 52 53 00 52 45 53 LIST,FOLDERS.RES
```

Word format:

```
##dw 1F:0

001f:00000000 5057 4f5f 4a42 4148 444e 454c 313d 3737
001f:00000010 3131 0030 3555 5245 495f 494e 433d 5c3a
001f:00000020 534f 5c32 534f 2e32 4e49 0049 5953 5453
001f:00000030 4d45 495f 494e 433d 5c3a 534f 5c32 534f
001f:00000040 5332 5359 492e 494e 4f00 3253 535f 4548
001f:00000050 4c4c 433d 5c3a 534f 5c32 4d43 2e44 5845
001f:00000060 0045 5541 4f54 5453 5241 3d54 4154 4b53
001f:00000070 494c 5453 462c 4c4f 4544 5352 5200 5345
```

Double-word format:

```
##dd 1f:0

001f:00000000 4f5f5057 41484a42 454c444e 3737313d
001f:00000010 00303131 52455355 494e495f 5c3a433d
001f:00000020 5c32534f 2e32534f 00494e49 54535953
001f:00000030 495f4d45 433d494e 534f5c3a 534f5c32
001f:00000040 53595332 494e492e 32534f00 4548535f
001f:00000050 433d4c4c 534f5c3a 4d435c32 58452e44
001f:00000060 55410045 54534f54 3d545241 4b534154
001f:00000070 5453494c 4c4f462c 53524544 53455200
```

# DA - Display Memory in ASCII Format



Display a range of memory from a given address in ASCII format.

**Syntax:**

```
DA
    addr
    Ln
```

See the [D command](#) for a full description.

# DB - Display Memory in Byte Format



Display a range of memory from a given address in Byte format.

**Syntax:**

```
DB
    addr
    Ln
```

See the [D command](#) for a full description.

## DW - Display Memory in Word Format



Display a range of memory from a given address in Word format.

**Syntax:**

```
DW          ~
            addr      Ln
```

See the [D command](#) for a full description.

## DD - Display Memory in Double-word Format



Display a range of memory from a given address in Double-word format.

**Syntax:**

```
DD          ~
            addr      Ln
```

See the [D command](#) for a full description.

## DG - Display Global Descriptor Table



Display entries from the [Global Descriptor Table](#).

**Syntax:**

```
DG          ~
DGA         s      Ln
```

**Parameters:**

*(Default)*                      Display valid GDT entries only.

**A**

Display all GDT entries including invalid descriptors.

**s**

Display descriptor for selector number **s**.

#### Notes

Since bit 2 of the selector determines whether the descriptor is local or global the correct table entry will be displayed regardless of whether the **DG** or **DL command** is used. If an **LDT** descriptor is specified then the following message is displayed:

LDT

The requestor priority level (RPL) bits (bits 0 and 1 of the selector) are ignored by **DG**. Thus: **DG 8** displays the same information as **DG 9**, **DG a** and **DG b**.

If the **s** parameter is omitted then the entire **GDT** is displayed.

**L/n**

The number of descriptor entries to display from and including selector **s**. The default is to display one descriptor entry.

#### Results & Notes:

One or more descriptor table entries are displayed. An example display follows:

```
##dga
0000 Invalid Bas=00000000 Lim=00000000 DPL=0 NP
0008 Invalid Bas=00000000 Lim=00000000 DPL=0 NP
0010 TSS32 Bas=ffe05dfc Lim=00000067 DPL=0 P B
0018 Data Bas=ffe00150 Lim=000003ff DPL=0 P RW A UV
0020 Data Bas=ffe4a000 Lim=000003ff DPL=0 P RW A UV
0028 LDT Bas=7ab27000 Lim=0000ffff DPL=0 P
0030 Data Bas=ffde08a4 Lim=0000575b DPL=0 P RW ED A UV
003b Data Bas=7c38ba8c Lim=00000073 DPL=3 P RW
0040 Data Bas=ffe49400 Lim=000003bf DPL=0 P RW UV
004a Data Bas=00000000 Lim=1bffffff DPL=2 P RW A G4k BIG UV
0053 Data Bas=00000000 Lim=1bffffff DPL=3 P RW A G4k BIG UV
005a Code Bas=00000000 Lim=1bffffff DPL=2 P RE C A G4k C32 UV
0063 Data Bas=00000000 Lim=1bffffff DPL=3 P RW G4k BIG UV
006b Data Bas=00000000 Lim=1bffffff DPL=3 P RW A G4k BIG UV
```

For a detailed explanation of the descriptor table entry format see [Descriptor Table Entry Format](#)

## Descriptor Formats

The Kernel Debugger and Dump Formatter format descriptor table entries in either of two forms depending on whether the descriptor describes a segment of memory or a [gate](#):

```
dddd type Bas=bbbbbbb Lim=lllllllll DPL=p flags
```

```
dddd type Sel:Off=ssss:oooooooo DPL=p flags
```

Each of these fields has the following meaning:

**dddd**

Descriptor number

**type**

Descriptor type. The following are defined:

<i>Type</i>	<i>Type Numbers</i>	<i>Description</i>
Code	-	Code segment
Data	-	Data segment
Invalid	0 or 8	Invalid descriptor
TSS	1 or 3	Available or Busy 80286 TSS
LDT	2	system descriptor for an LDT
CallG	4	Call Gate
TaskG	5	Task Gate
IntG	6	80286 Interrupt Gate
TrapG	7	80286 Trap Gate
Reserve	10 or 13	Reserved descriptor types
TSS32	9 or 11	Available or Busy Intel486 CPU TSS
CallG32	12	Inter486 CPU Call Gate
IntG32	14	Intel486 CPU Interrupt Gate
TrapG32	15	Intel486 CPU Trap Gate

**Bas=bbbbbbbb**

Segment base address.

**Lim=////////**

Segment limit address.

**DPL=p**

Descriptor priority level. Only 0, 2 and 3 are used in OS/2.

**Sel=ssss:Off=00000000**

**selector:offset** transfer address for a task, interrupt, trap or call gate descriptor.

**flags**

Interpretation of the various descriptor flags. The following abbreviations are used:

<i>Flag</i>	<i>Bits</i>	<i>Description</i>
NP	~15	Not present
P	15	Present
RW	9	Read/Write data segment
RO	~9	Read-only data segment
ED	10	Expand-down data segment
C	10	Conforming code segment
G4k	23	4K granularity segment limit
BIG	22	32-bit Stack offsets (ESP) used as a stack segment. (No meaning when used as a data segment).
C32	22	32-bit operands and data

		sizes used by default with this code segment
RES	21	reserved
UV	20	Available bit. Used in OS/2 to indicate a <a href="#">UVIRT</a> mapping.
WC=w	0	Word count of a 16-bit call gate
DWC=w	0	Double-word count of a 32-bit call gate
RE	9	Read/Execute code segment
EO	-9	Read-only code segment
A	8	Code or Data segment accessed
NB	-	TSS/TSS32 not busy (available)
B	-	TSS/TSS32 busy

#### Notes

The bit offsets given above are relative to the second double-word of the descriptor viewed as 2 double-words. **The INTEL programmer's Reference** shows the descriptor format as a quad-word, but uses the same offsets specified above.

See the **INTEL Pentium User's Reference** or the **INTEL x86 Programmer's References** for further information.

## DI - Display Interrupt Descriptor Table



Display entries from the [Interrupt Descriptor Table](#).

#### Syntax:

```
DI
DIA          i          ~
                Ln
```

#### Parameters:

*(Default)*

Display valid IDT entries only.

**A**

Display all IDT entries including invalid descriptors.

**/**

Display descriptor for [interrupt vector](#) **/**.

**Ln**

The number of descriptor entries to display from and including selector **/**. The default is to display one descriptor entry.

**Results & Notes:**

One or more descriptor table entries are displayed. An example display follows:

```
##dia
0000  TrapG32 Sel:Off=0170:fff47e64      DPL=0  P
0001  IntG32  Sel:Off=0170:fff47f10      DPL=3  P
0002  TaskG   Sel:Off=1e38:00000000      DPL=0  P
0003  IntG32  Sel:Off=0170:fff480cc      DPL=3  P
0004  TrapG32 Sel:Off=0170:fff48158      DPL=3  P
0005  TrapG32 Sel:Off=0170:fff48164      DPL=0  P
0006  TrapG32 Sel:Off=0170:fff48170      DPL=0  P
0007  TrapG32 Sel:Off=005a:1a090911      DPL=0  P
0008  TaskG   Sel:Off=0088:00000000      DPL=0  P
0009  TrapG32 Sel:Off=0170:fff48258      DPL=0  P
000a  TrapG32 Sel:Off=0170:fff48268      DPL=0  P
000b  TrapG32 Sel:Off=0170:fff48270      DPL=0  P
000c  TrapG32 Sel:Off=0170:fff48278      DPL=0  P
000d  TrapG32 Sel:Off=0170:fff48280      DPL=0  P
000e  TrapG32 Sel:Off=0170:fff4853c      DPL=0  P
000f  TrapG32 Sel:Off=0170:fff48544      DPL=0  P
0010  TrapG32 Sel:Off=0170:fff4854c      DPL=0  P
```

For a detailed explanation of the descriptor table entry format see [Descriptor Table Entry Format](#)

# DL - Display the Current Local Descriptor Table



Display entries from the [Local Descriptor Table](#). of the default [thread slot](#). See the [.S command](#) for information of changing the default thread slot.

**Syntax:**

```
DL
DLA      s      Ln
DLP
DLS
DLH
```

**Parameters:**

- (Default)* Display valid LDT entries only.
- A Display all LDT entries including invalid descriptors.
- P Obsolete option. Was used to display only valid private arena LDT descriptors where bits 3 and 4 of the selector number are 0.
- S Obsolete option. Was used to display only valid shared arena LDT descriptors where bits 3 and 4 of the selector number are non-zero.
- H Obsolete option. Was used to display only huge segment LDT descriptors.
- s Display descriptor for selector number *s*.

**Notes**



Since bit 2 of the selector determines whether the descriptor is local or global the correct table entry will be displayed regardless of whether the **DL** or **DG command**. is used. If an **GDT** descriptor is specified then the following message is displayed:

GDT

The requestor priority level bits (bits 0 and 1 of the selector) are ignored by **DL**. Thus **DL 7** displays the same information as **DL 6**, **DL 5** and **DL 4**.

If the **s** parameter is omitted then the entire **LDT** is displayed.

**L/n**

The number of descriptor entries to display from and including selector **s**. The default is to display one descriptor entry.

### **Results & Notes:**

One or more descriptor table entries are displayed. An example display follows:

```
##dl
0007 Data Bas=7ab27000 Lim=0000ffff DPL=3 P RO
000f Code Bas=00010000 Lim=000005ff DPL=3 P RE
0017 Data Bas=00020000 Lim=0000005b DPL=3 P RW
001f Data Bas=00030000 Lim=0000fa1f DPL=3 P RW A
0027 Data Bas=00040000 Lim=00000276 DPL=3 P RW A
002f Data Bas=00050000 Lim=00000fff DPL=3 P RW
0036 Data Bas=00060000 Lim=00003fff DPL=2 P RW A
003f Data Bas=00070000 Lim=00000fff DPL=3 P RW A
0047 Data Bas=00080000 Lim=00000fff DPL=3 P RW
004f Data Bas=00090000 Lim=0000ffff DPL=3 P RW A
0056 Code Bas=000a0000 Lim=00000af7 DPL=2 P RE C
005f Data Bas=000b0000 Lim=0000ffff DPL=3 P RW
```

For a detailed explanation of the descriptor table entry format see [Descriptor Table Entry Format](#)

## DP - Display Page Directory and Table Entries



Display entries from the page tables of the default [thread slot](#). See the [.S command](#) for information on changing the default thread slot.

### **Syntax:**

```
DP
DPD addr
DPA Ln
```

### **Parameters:**

**A** Display both page table and page directory entries. This is the default.

**D** Display only page directory entries.

**addr** The [linear or virtual address](#) whose page directory and table entries are to be displayed. If not specified **DP** displays the entire page directory and its page tables.

An [address expression](#) may be specified.

## L/n

The number of page table entries to display starting with the entry for *addr*. The default is to display the all page table entries from this entry assigned to *addr*.

**Note:** Due to a bug in some versions of the Kernel Debugger an extra zero is required for this parameter.

## Results & Notes:

One or more page and directory table entries are displayed. An example display follows:

DP %90000 L50

linaddr	frame	pteframe	state	res	Dc	Au	CD	WT	Us	rW	Pn	state
%00090000*	012f3	frame=012f3	2	0	D	A			U	W	P	resident
%00090000		vp id=00a76	0	0	c	u			U	W	n	pageable
%000a0000	000b8	vp id=000b8	1	0	D	u			U	W	n	uvirt
%000b0000	00888	frame=00888	0	0	D	A			U	W	P	pageable
%000c0000		vp id=00b8f	0	0	c	u			U	W	n	pageable
%000d0000		vp id=00b92	0	0	c	u			U	W	n	pageable
##												

Output from the **DP** command is presented in tabular form. Each of the columns shown is described as follows:

### linaddr

Linear address of virtual memory whose page directory and table entries are being formatted. Those lines corresponding to directory entries have an \* flag suffixed to the linear address. Page table entries for a given directory entry are formatted following the directory entry.

In the example above linear address **%90000** has its page table located in physical frame **12f3**, that is at physical **%%12f3000**. The page table entry corresponding to virtual memory at **%90000** is described in the second line. Each of the following lines are consecutive entries from page table **12f3**.

### frame

The real storage frame number that contains either the page table (\* suffix to **linaddr**) or page frame corresponding to the **linaddr**. If this field is blank then the frame has been discarded. If it contains a frame number then the contents are still valid even though the page table entry no longer points to a page frame. See **pteframe** field for further discussion.

### pteframe

For table entries with the present bit set the this field shows the page frame number pointed to by this table entry. This is shown as **frame=ffff**. Use the frame number with the **.MP command** to obtain information on allocation and ownership of this this frame of real storage.

For decommitted pages the table entry contains the **Virtual Page ID**. This is shown as **vp id=vvvvv**. Use the **.MV command** with the virtual page Id to obtain information on allocation and ownership of this memory.

### Notes

The **vp id** is not valid to use with **.MV** if the **state** of the table entry is **uvirt**.

If the frame has been decommitted but the **frame** field still shows a frame number then the frame contents are still valid for reclaiming without a page-in operation from the swap file. The corresponding virtual page will be queued from the idle list. See **.MV** and **.MP** commands for more information on page management.

### state

State information is stored in the **available** bits (9 - 11) of the page table entry. These are interpreted on the right-hand end of the display. The following values may appear:

State	Value	Description
pageable	0	Storage may be paged-out to the swap file
uvirt	1	Physical to virtual mapping reservation only.
resident	2	Non-pageable fixed storage
uvirt	3	Physical to virtual

mapping reservation only.

<b>Res</b>	Reserved page table entry bits. Should always be zero
<b>Dc</b>	Set to <b>D</b> is the page is dirty, otherwise <b>c</b> (clean).
<b>Au</b>	Set to <b>A</b> if the page has been accessed, otherwise <b>u</b> (unaccessed).
<b>CD</b>	Set to <b>CD</b> is the TLB cache-disable bit is set, else blank.
<b>WT</b>	Set to <b>WT</b> is the TLB cache write-transparent bit is set, else blank.
<b>Us</b>	Set to <b>U</b> if the page is for user storage, otherwise <b>s</b> (supervisor).
<b>rW</b>	Set to <b>r</b> is the page is read-only, otherwise <b>W</b> (writeable).
<b>Pn</b>	Set to <b>P</b> if the page is present, otherwise <b>n</b> (not present).

**Note:**

The Dump Formatter does not format page directory entries correctly. For page directory entries only the **frame** field is correct. The remaining fields are taken from the first PTE of the page table associated with the page directory. This problem has been fixed from fix pack 36 for Warp 3.0 and fix pack 7 for Warp 4.0.

Refer to the following for more information on page and memory management:

[.M family of Kernel Debugger and Dump Formatter commands.](#)

**Intel Pentium User's Guide** and **Intel x86 Programmer's Reference**.

---

## DT - Display a Task State Segment



Format a [task state segment](#).

**Syntax:**

```
DT                               ^
                                addr
```

**Parameters:**

*addr*

The address of the task state segment to be formatted. If not specified then the current TSS pointed to by the **TR** (task register) is used.

**Results & Notes:**

The TSS is formatted as follows:

```
##dt 10:0
eax=00000000 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000 edi=00000000
eip=00000000 esp=00000000 ebp=00000000 iopl=0 -- -- -- nv up di pl nz na po nc
cs=0000 ss=0000 ds=0000 es=0000 fs=0000 gs=0000 cr3=001d1000
ss0=0030 esp0=000066fc ssl=0000 esp1=00000000 ss2=0866 esp2=00001000
ldtr=0028 link=0000 tflags=0000 i/o map=dfff
ports trapped: 0-ffff
```

Each of the fields displayed has the following meaning:

**Note:**

Some of the **TSS** fields are set at task creation and other when a task switch occurs.

<b>eax=</b>	Saved <b>EAX</b> register when a task switch occurs.
<b>ebx=</b>	Saved <b>EBX</b> register when a task switch occurs.
<b>ecx=</b>	Saved <b>ECX</b> register when a task switch occurs.
<b>edx=</b>	Saved <b>EDX</b> register when a task switch occurs.
<b>esi=</b>	Saved <b>ESI</b> register when a task switch occurs.
<b>edi=</b>	Saved <b>EDI</b> register when a task switch occurs.
<b>eip=</b>	Saved <b>EIP</b> register when a task switch occurs.
<b>esp=</b>	Saved <b>ESP</b> register when a task switch occurs.
<b>ebp=</b>	Saved <b>EBP</b> register when a task switch occurs.
<b>iopl=</b>	Saved <b>EGLAGS</b> iopl and flag settings when a task switch occurs. See <a href="#">.R command</a> for an explanation of the flag abbreviations.
<b>cs=</b>	Saved <b>CS</b> register when a task switch occurs.
<b>ss=</b>	Saved <b>SS</b> register when a task switch occurs.
<b>ds=</b>	Saved <b>DS</b> register when a task switch occurs.
<b>es=</b>	Saved <b>ES</b> register when a task switch occurs.
<b>fs=</b>	Saved <b>FS</b> register when a task switch occurs.
<b>gs=</b>	Saved <b>GS</b> register when a task switch occurs.
<b>cr3=</b>	<b>CR3</b> register at task creation.
<b>ss0=</b>	<b>Note:</b> This provides the real address of the Page Directory Table, which never alters under OS/2.

	Ring 0 <b>SS</b> register used for ring 0 privilege transitions.
<b>esp0=</b>	Ring 0 <b>ESP</b> register used for ring 0 privilege transitions.
<b>ss1=</b>	Ring 1 <b>SS</b> register used for ring 1 privilege transitions.
<b>esp1=</b>	Ring 1 <b>ESP</b> register used for ring 1 privilege transitions. <b>Note:</b> Ring 1 is not used under OS/2.
<b>ss2=</b>	Ring 2 <b>SS</b> register used for ring 2 privilege transitions.
<b>esp2=</b>	Ring 1 <b>ESP</b> register used for ring 2 privilege transitions.
<b>ldtr=</b>	<b>LDTR</b> register at task creation.
<b>link=</b>	<b>TR</b> register value of previous nested task's <b>TSS</b> .
<b>tflags=</b>	The debug trap bit for this task.
<b>i/o maps=</b>	Offset to the I/O permission map from the beginning of this TSS. <b>Note:</b> It is permissible for the i/o map offset to point beyond the TSS segment. This signifies that no I/O permissions are granted and all ports will be trapped.
<b>ports trapped:</b>	List the range of I/O port addresses that will generate traps if accessed by this task.

#### Notes:

For performance reasons hardware implemented task switching is used only in a limited way in OS/2. **TSSs** defined by OS/2 include:

- Protect mode code (**TSS** selector **10**)
- Virtual DOS Machines
- Non Maskable Interrupt handling (trap 2, **TSS** selector **1E38**)
- Double Fault handling (trap 8, **TSS** selector **88**)

All protect-mode processes run under a common top-level task using selector 10 as the **TSS** selector.

The **sel<sub>tss</sub>** (**PTDA** +0x2f0) field of the **PTDA** records the top-level task's **TSS** selector used by a given process, thus may be used to find the **TSS** selector for Virtual DOS Machines.

Refer to **Intel Pentium User's Guide** and **Intel x86 Programmer's Reference** for more information on the Task State Segment and Hardware architected multi-tasking.

-----

## DX - Display the 286 LoadAll Buffer



Formats a the 286 LoadAll buffer from physical address **%%800** in memory.

#### Syntax:

DX

**Parameters:**

None.

**Results & Notes**

This command applies to the Intel 286 processor and is now obsolete. The results are meaningless.

---

## E - Enter Data Into Memory



Enter data into a memory location.

**Syntax:**

```
E          addr          value
                        string
```

**Parameters:**

***addr***

The address of the memory location to be changed. If not specified this defaults to **DS:00000000** where **DS** is established by the most recent register display. See

An [address expression](#) may be specified. [R](#) and [.R](#) commands for information on establishing default addresses.

***value***

A numerical byte value to be entered into memory. One or more values may be specified separated commas or blanks. These may be mixed with ***"string"*** values.

***string***

A character sting enclosed in quotes. Each character is treated as a byte value and entered into memory separately, no terminating 0x00 value is stored. No folding of characters to upper or lower case occurs. One or more strings may be specified separated by commas or blanks. These may be mixed with numerical ***values***.

**Results & Notes:**

If memory is present values are entered into storage otherwise an **Invalid Address** message is generated. If this should happen, valid storage may be paged into memory by means of the [.I command](#).

If no ***value*** or ***string*** parameter is specified the Kernel Debugger prompts the user a byte at a time for replacement values by displaying the original value followed by a colon. In prompt mode, the user may proceed as follows:

type a replacement byte value in hexadecimal, or

accept the original value and move on to the next location by pressing the space-bar, or

back up to the previous location by entering a - (minus) character, or

terminate prompt mode by pressing carriage return (with or without a replacement value).

---

## F - Fill Memory With Repeated Data



Fill memory with repeated data.

**Syntax:**

```
F      addr      Ln      value
                        string
```

**Parameters:**

- addr**  
The address of the memory location to be changed.  
An [address expression](#) may be specified.
- Ln**  
The number (*n*) of bytes to fill with data.
- value**  
A numerical byte value to be entered into memory. One or more values may be specified separated commas or blanks. These may be mixed with *"string"* values.
- string**  
A character sting enclosed in quotes. Each character is treated as a byte value and entered into memory separately, no terminating 0x00 value is stored. No folding of characters to upper or lower case occurs. One or more strings may be specified separated by commas or blanks. These may be mixed with numerical *values*.

**Results & Notes:**

The list of *values* and *strings* is repeated up to the length **Ln** and used to fill memory at the specified address. If the fill data is shorter than the length then it is repeated, if it is longer it is truncated.

If memory is present the storage is updated otherwise an **Invalid Address** message is generated. If this should happen, valid storage may be paged into memory by means of the [.I command](#).

-----

## G - Go



Cause execution to continue from a given point and optionally set 1 or more 'go' breakpoints.

**Syntax:**

```
G      =      start-addr      break-addr
GS
GT
```

**Parameters:**

- (Default)*  
Continue execution from the current **CS:EIP**.

## S

The *go-special* command causes the high-resolution time interval to be recorded from the point **GS** command is issued to the point that the Kernel Debugger is re-entered as the result of a break-point *firing*.

### Notes

No account is taken of the Kernel Debugger overhead when calculating the time interval.

When the Kernel Debugger re-enters, for whatever reason, the interval timer is cancelled until another **GS** command is executed.

If the reason for entry is for reasons other than the *firing* of a *sticky* or *go breakpoint* then in addition to cancelling the interval timer no time message displayed.

## T

This option causes the Kernel Debugger's trap vector handlers to be removed temporarily from the **IDT** and the system's re-instated until after then next instruction has executed. After execution of the next instruction the Kernel Debugger's **V commands** are re-instated.

This is a convenience option that saves manually unhooking a Kernel Debugger trap vector handlers from the **IDT**. using a command sequence similar to:

```
VC n
T
VS n
G
```

### *start-addr*

The address from which execution is to continue. This must be a valid address for the current **context**. If *start-addr* is omitted then execution continues from the current **CS:EIP**, as shown by the **R command**.

**Warning:** Be very careful to ensure that the start address is valid for the privileged level and addressability of the code and data selectors in use. If the Kernel Debugger attempts to load a segment register that is invalid the system may trap in the debugger code.

### *break-addr*

Up to ten *go breakpoints* may be specified. These are temporary breakpoints set in addition to any *sticky* breakpoints set by the **B commands**. When the Kernel Debugger is next entered, for whatever reason, all *go* breakpoints are cleared.

If *break-addr* is omitted then the system continues execution until:

A fatal exception occurs

An Internal Processing Error (IPE) occurs.

A *sticky* breakpoint fires

A non-maskable interrupt occurs

An **INT 3** instruction is executed

The user enters **Ctrl-C** from the debugging console.

### Results & Notes:

The system continues execution until the Kernel Debugger is re-entered. If the reason for entry is other than a breakpoint *firing* then the **R command** is automatically executed followed by one of the following command prompts:

- > (signifies a command prompt in real mode)
- # (signifies a command prompt in protect mode with paging disabled)
- (signifies a command prompt in V86 mode with paging disabled)
- ## (signifies a command prompt in protect mode with paging enabled)
- (signifies a command prompt in V86 mode with paging enabled)

If an error situation caused entry to the Kernel Debugger then a diagnostic message may be generated by the failing code writing directly to the Kernel Debugger's communications port.



If entry was caused by a Kernel Debugger trap handler receiving control then a message from the trap handler will be displayed. See the [V command](#) for details.

If a breakpoint caused the Kernel Debugger to receive control then commands associated with the breakpoint that fired will execute. See the [B commands](#) for details.

If a *go-special* was interrupted by a breakpoint firing then the following message appears before any output associated with the breakpoint:

```
Go Time (tics) = 017fb (hex) = 5145 (uSec)
```

This shows the time interval in both timer-ticks and equivalent number of micro-seconds.

-----

## H - Hex Arithmetic



Display the sum, difference, product, quotient and remainder of two absolute expressions.

### Syntax:

```
H      abs-expr1      abs-expr2      \
```

### Parameters:

#### *abs-expr1*

An expression that resolves to a [simple numeric](#) value using any of the [expression evaluator operators](#). Symbols of [absolute values](#) may be specified in the expression, but symbols of relocatable addresses may not.

#### *abs-expr2*

An expression that resolves to a [simple numeric](#) value using any of the [expression evaluator operators](#). Symbols of [absolute values](#) may be specified in the expression, but symbols of relocatable addresses may not.

### Results & Notes:

Each of the expressions is evaluated. If either does not resolve to a simple numeric value then the message:

```
Expression error
```

is displayed.

Having resolve each of the expressions then the sum, difference, product and quotient of the pair is displayed as in the following examples:

```
-----
##h 2 3
+0005 -ffff *0006 0000 /0000 0002
#h 10t 5
+000f -0005 *0032 0000 /0002 0000
##h 7fff 5
+8004 -7ffa *7ffb 0002 /1999 0002
## 5*4 2*3
+001a -000e *0078 0000 /0003 0002
##h bmp_segsize 5
+0017 -000d *005a 0000 /0003 0003
##h
-----
```

### **Notes**

Calculations are performed using 16-bit signed arithmetic.

The operation performed is shown prefixing the result.

The product is shown as a two word value, the low word followed by the high.

The division is shown as two words, the quotient followed by the remainder.

In the last example, **bmp\_segsize** is an absolute symbol of value **0x0012** defined in map **OS2KRNL**.

Evaluation of complex expressions involving relocatable addresses may be done using the [? command](#).

-----

## I - Input From an I/O Port



Input a byte of data from a 16-bit I/O Port

### Syntax:

```
I      port
```

### Parameters:

*port*

A 16-bit I/O port address. This may be specified as a simple numeric expression.

### Results & Notes:

The byte of data is read from the requested I/O port and displayed in hexadecimal at the console. For example:

```
-----  
##I 2f8  
0d  
-----
```

See also the [O command](#) for related information.

-----

## J - Execute Commands Conditionally



Conditionally execute one of two lists of commands depending on whether an expression evaluates to **TRUE** (non-zero) or **FALSE** (zero).

### Syntax:

```
J      expression  
      " cmd1  
      " cmd1  
      ; cmd2
```

### Parameters:

### *expression*

An expression that resolves to either a simple numeric value or an address using any of the [expression evaluation](#) operators. [Symbols of addresses](#) and [symbols of absolute values](#) may be specified.

### *cmd1*

Specifies a command to be executed if the *expression* evaluates to **TRUE** (non-zero). More than one command may be specified if each is separated by a semi-colon and the entire command list is enclosed in single or double quotes.

If *cmd1* is omitted, control is returned to the debugging console when the *expression* is **TRUE**.

### *cmd2*

Specifies a command to be executed if the *expression* evaluates to **FALSE** (non-zero). More than one command may be specified. Each *cmd2* must be prefixed by a semi-colon, even if only one is specified. Quotes are not required to encompass a list of

If *cmd2* is omitted, control is returned to the debugging console when the *expression* is **FALSE**. *cmd2* commands.

## Results & Notes:

If the expression resolves to one of the following forms, it is considered to be **FALSE** false:

```
0
0:0
&0:0
%0
%%0
```

Any other resolution is regarded as **TRUE**.

The **J** command is primarily intended to be used with the **BP** and **BR** commands to enable conditional [breakpoints](#) to be defined.

Examples of this usage are:

```
-----
BP #f:12d5 "J ax!=10t;g"

BP #f:12d5 "J ax==10t;g"

BP SchedNextRet "J wo(Tasknumber)==8, '.p*;.r';g"

BP DOSOPEN "J wo(Tasknumber)==32, 'da #(wo(ss:sp+20)):(wo(ss:sp+1e));g';g"
-----
```

The first example shows a breakpoint set at address **#f:12d5**. When this breakpoint *fires* the **J** command tests the condition **AX** register not equal to decimal 10. If this is true, the **G** command is executed. Since no *cmd2* is specified the **J** command returns control to the debugging console when the condition is false (**AX** equal to decimal 10).

The second example is has the same effect as the first but is implemented by testing the logically opposite condition.

The third example shows one method of stopping the system when a thread switch to a particular thread slot has just occurred. In this case the debugging console gains control when thread slot 8 is selected, whereupon **.p\*** and **.r** are automatically executed. The breakpoint **SchedNextRet** is one of two exit points from the scheduler (**\_tkSchedNext**). The other, **SchedNextRet2** is taken when the same thread slot is selected for re-dispatch. The global variable **Tasknumber** contains the current and therefore out-going slot number on entry to the scheduler; and in-coming slot number on exit from the dispatcher.

### **Note:**

The kernel calls one of the **KMExitKmode** routines before giving control to user code. During this kernel exit processing the **Resched** and ( **TCB** and **PTDA**) force flags are checked again and if set the scheduler/dispatcher sequence is invoked. It is possible therefore, that even though a thread is selected to run, and achieves **run** state, it is put back on the ready queue before being given any user processing time.

The fourth example illustrates a method of tracing resources that are opened by a specific thread slot (in this case slot 32) without giving control to the debugging console. **DOSOPEN** is the kernel's entry point for open processing. At this point words **0x0f** and **0x10** contain the offset and selector that points to the resource name.

-----

## K - Display Stack Trace from Address



Display the stack-trace from a given stack frame address.

### Syntax:

```
K
KS      stack-frame      selector:offset
KB
```

### Parameters:

**K** Display stack frame trace assuming the default operation size from the code descriptor specified by *selector:offset*

**KS** Display frame trace assuming an operation size of 16-bits (small-model).

**KB** Display frame trace assuming an operation size of 32-bits (big-model).

#### *stack-frame*

Address of the starting stack-frame. If not specified then this defaults to the current **SS:EBP** or **SS:BP** as set by the last register display.

See the [R](#) and [.R](#) commands for information on changing the default register values.

An [address expression](#) may be specified.

#### *selector:offset*

The *selector:offset* address of the code that is in effect when the starting *Stack-frame* address was created. If not specified this defaults to the current **CS:EIP** or **CS:IP** as displayed by the

See the [R](#) command.

The code selector associated with this address is used for two purposes:

- 1 To determine the default operand size in effect from the code segment descriptor.
- 2 To attempt to distinguish between near and far calls at the starting *stack-frame* address.

### Results & Notes:

The **K** command displays the stack trace, threading through the **BP** or **EBP** chain until either an invalid chain pointer is encountered or the command is interrupted by the user. For each stack-frame, the return address and for parameter words or double-words are displayed. The symbol associated with the return address is displayed after the parameter words. An example is given below:

```
-----
##.S 8
##.R
eax=c7c00000 ebx=00000014 ecx=003acd7 edx=0000aff7 esi=00030bff edi=00030000
eip=0000272d esp=0003f8b8 ebp=0003f8c0 iopl=2 -- -- -- nv up ei ng nz na pe nc
cs=d02f ss=001f ds=aff7 es=be47 fs=150b gs=0000 cr2=1701d000 cr3=001d9000
doscall1:CONFORM16:postDOSSEMWAIT:
d02f:0000272d c9 leave
##K SS:BP CS:IP

bdef:0000711e ffff ffff 06d6 0a23 SEEPSMQ + 67
bdef:0000e1df ffff ffff 0bff f91c GETMSGNOINPUT + 4a
belf:00000271 8001 ffff 0000 0000 THK16_CALLUSERTHUNKPROC + 12
belf:00000003 05ae 0001 0001 0003 THUNKTOINITMOVECURSOR + 3
0020:00000003 0001 0001 0000 0010
0020:00000000 02af 1a03 0197 0000
##-----
```

### **Notes**

1. The **K** command is insensitive to unconventional use of the stack, such as where subroutine returns are effected explicitly by setting the stack pointer and jumping back to the calling code or in optimised code where the **EBP** or **BP** registers are not used as stack-frame pointers.

Such possibilities exist within the system when for example the kernel returns to user code and also within some Presentation Manager components.

2. No attempt is made to trace correctly through [thunking](#) layers where the default operand size changes.
3. The stack trace is insensitive to any explicit segment operand overrides that may be active.
4. No attempt is made to examine the descriptor of the **SS** register to determine whether **EBP** or **BP** should be used. In much 32-bit code both the 16-bit and 32-bit data descriptors are created by the system for calls to 16-bit subroutines.

In the example above the stack-frame address has been explicitly overridden to use **BP** since the 16-bit stack selector (**1f**) is in effect rather than the 32-bit **53** selector.

5. Unlike the default stack-frame address the default code *selector:offset* is taken from the register values on entry to the Kernel Debugger.

#### Warning:

In consequence of these points it is recommended that the *stack-frame* and code *selector:offset* addresses be explicitly coded when using the **K** command, as in the example above. In addition the stack trace should be verified with a [memory dump](#) of the stack.

-----

## L - List Maps, Groups and Symbols



List maps, groups and symbols from loaded symbol files.

See the [W command](#) for related information.

#### Syntax:

```
L      A      ~
      M      map-name
      G

      S      ~
      addr

      N      ~
      addr
      symbol
```

#### Parameters:

- A** List [absolute symbol](#) definitions for the specified *map-name* or for all active [maps](#).
- M** List all active [maps](#) or the status of the specified map.
- G** List [groups](#) defined in all active maps of the specified map.

## N

When *addr* is specified this option list the nearest [symbols](#) to the address. If an exact match is found that symbols is listed otherwise the nearest symbol before and after *addr* is listed.

When *symbol* is specified then the address, map and group corresponding to the symbol is listed.

If neither *addr* nor *symbol* is specified then the default disassembly address is assumed. See the [.R](#) and [U](#) commands for related information.

## S

List all symbols defined in the group that encompasses *addr* for all active maps. If *addr* is not specified then the value of **CS:EIP** on entry to the debugger is assumed, as displayed by the [R command](#).

### *map-name*

Specifies the link edit map name from which information is to be displayed.

### *addr*

Specifies an explicit [address expression](#)..

### *symbol*

Specifies a publicly defined [symbol](#) name from a program source code.

## Results & Notes:

Symbol maps are obtained from symbol files (\*.SYM), which are generated using the **linkage editor** and the **MAPSYM** utility. Under the Kernel Debugger they are loaded from the same directory as their corresponding load module when that is loaded by the system. When this happens the **Symbols linked (*map-name*)** message appears. When a load module is deleted from the system, its map is removed and the message **Symbols unlinked (*map-name*)** appears.

Under the Dump Formatter symbol files are loaded for each [MTE](#) in the dump, during initialisation, from the current directory (usually the directory the Dump Formatter is running from).

Under the Dump Formatter conforming segments are not checked. Thus a ring 2 selector:offset address may not be recognised, whereas the ring 3 selector is. If **LN** does not find a symbol for a ring 2 selector, try specifying the same selector with the ring 3 RPL specified. For example, specify **d0fe:1234** as **d0ff:1234**.

Under the Dump Formatter **LN** does not check equivalences of the selector:offset and linear forms of an address. Therefore it may be necessary to apply the [CRMA](#) to an address if **LN** fails to find any near symbols.

Loaded symbol maps be **active** or **inactive**, depending on whether the corresponding load module is (potentially) active in the current context. In the case of private executable modules erroneous symbolic information may be associated with a private storage location. For this reason maps may be manually activated and removed using the [W command](#).

Maps for WINOS2 and WINDOWS components are supported under the Kernel Debugger only. These are automatically activated and deactivated according to whether the Kernel Debugger default [thread slot](#) is a WINDOWS or WINOS2 environment.

Output from each of the **L** subcommands is more or less self explanatory. Examples follow:

```
-----
##1a
cmd:
9876 __acrtmsg
9876 __acrtused
d6d6 __aDBused
d6d6 __aDBdoswp
-----
```

List absolute symbols defined in cmd.exe and their associated constants.

```
-----
##1m
cmd is active
kernel [0040, 003f]
minxobj is active
wpprint is active
nwiapi is active
rexxinit is active
pmmle is active
fka is active
ibmdevr is active
ibmvgar is active
pmpre is active
```

os2krnl is active

List current map status.

**Note:**

The Windows **Kernel** is not active, but loaded in thread slots **40** and **3f**. The additional active slot number information is only provided with **WINDOWS** and **WINOS2** environment map files.

```
-----
##lg cmd
cmd:
000f:00000000 _TEXT1
0017:00000000 _TEXT3
001f:00000000 DGROUP
-----
```

List segment groups defined in cmd.exe and their associated addresses.

```
-----
##ln %20000
%00020000 cmd:_TEXT3:_eChcp
##ln _tkshednext
%fff4521c os2krnl:DOSHIGH32CODE:_tkSchedNext
##ln
0170:fff44695 os2krnl:DOSHIGH32CODE:HaltInst + 1
0170:fff44787 postSchedNext - f1
-----
```

List near symbols and their associated addresses.

**Note:**

In this example three uses of **LN** are shown.

1. Address **%20000** is shown to coincide with **\_eChcp** in the **\_TEXT3** group of **CMD.EXE**
2. Symbol **\_tkshednext** is shown to be at address **%fff4521c** in the **DOSHIGH32CODE** of **OS2KRNL**.
3. The current **CS:EIP** is at **+1** byte from **HaltInst** in group **DOSHIGH32CODE** of module **OS2KRNL** and **-f1** bytes before **postSchedNext** in the same group and module.

```
-----
##ls %fff3f500
%fff3f4a4 DevWOHandle
%fff3f4ac g_CodeLockProc
%fff3f4b1 CodeLockProc
%fff3f5a4 g_CodeUnlockProc
%fff3f5a9 CodeUnlockProc
%fff3f614 _FSAbortVDM
%fff3f62c FS32IREAD
%fff3f638 FS32IWRITE
%fff3f644 w_Big32IO
%fff3f6c0 w_SetFileLocks
%fff3f6c8 w_ProtectSetFileLocks
.
.
.
-----
```

List symbols in the current group encompassing address %fff3f500

See the [W command](#) for related information.

# M - Move a Block of Data in Memory



Move a block of contiguous data from one memory location to another. This command guarantees to duplicate the source data even when source and destination overlap.

## Syntax:

```
M      source-addr      Ln      target-addr      \
```

## Parameters:

### *source-addr*

The source address of the memory location to be moved (copied).

An [address expression](#) may be specified.

### *Ln*

The number (*n*) of bytes to move.

### *target-addr*

Target address of the memory move operation.

An [address expression](#) may be specified.

## Results & Notes:

Memory content is copied from the source to the target address. If the source and target overlap then source will be updated, however the move operation is conducted from highest to lowest address or *vice versa* depending on whether the target address is higher or lower than the source, thereby guaranteeing a faithful copy of the original source.

If memory is present the storage is updated otherwise an **Invalid Address** message is generated. If this should happen, valid storage may be paged into memory by means of the [.I command](#).

-----

# O - Output to an I/O Port



Output a byte of data to a 16-bit I/O Port

## Syntax:

```
O      port      data      \
```

## Parameters:

### *port*

16-bit I/O port address

### *data*

A byte of data expressed numerically. This may be specified as a simple numeric expression.

## Results & Notes:



The byte is sent to the requested I/O port.

**Note:**

This command may be used to set the debugging communication port parameters from the Kernel Debugger initialisation command file (**KDB.INI**) as in the following example:

```
-----
Set COM2 DTR line (assume standard port assignment for COM2 that is, 2f8):
##O 2fc 1

Set COM1 DTR line (assume standard port assignment for COM1 that is, 3f8):
##O 3fc 1
-----
```

# P - PTrace Instruction Execution



Trace instruction execution within a single procedure. This command is very similar to the [T command](#), except that **CALL**, loop and string repeat instructions are traced as single instructions (even though allowed to execute correctly).

**Syntax:**

```
P
      N
      T      =      start-addr      count      \
```

**Parameters:**

(Default)

Trace instruction execution by single-stepping, treating **CALL** loop and string repeat instructions as single events.

**Note:**

Certain areas of the system are known to cause problems if traced. Attempts to trace these areas are intercepted by the Kernel Debugger. See below for further information.

**N**

Trace instructions but suppress the register display after each instruction is executed.

**T**

This option causes the Kernel Debugger's trap vector handlers to be removed temporarily from the [IDT](#) and the system's re-instated until after then next instruction has executed. After execution of the next instruction the the Kernel Debugger's [V commands](#) are re-instated.

This is a convenience option that saves manually unhooking a Kernel Debugger trap vector handlers from the [IDT](#). using a command sequence similar to:

```
VC n
P
VS n
```

**start-addr**

The address from which execution is to continue. This must be a valid address for the current context. If **start-addr** is omitted then execution continues from the current **CS:EIP**, as shown by the [R command](#).

**count**

The number of instructions to trace before re-entering the Kernel Debugger unless one of the following conditions is

encountered:

- A fatal exception occurs
- An Internal Processing Error (IPE) occurs.
- A 'sticky' breakpoint fires
- A non-maskable interrupt occurs
- An **INT 3** instruction is executed
- The user enters **Ctrl-C** from the debugging console.
- If omitted then *count* defaults to **1** instruction.

### Results & Notes:

The **Ptrace** commands trace the execution of machine instructions, and by default, display the current registers and next instruction to execute at each step. For the purposes of the displayed trace, the **CALL** instruction does not have the called routine traced, but tracing resumes on return. Loop and string repeat instructions are also treated as atomic entities with the instruction following the loop or repeat shown as the next to execute. **INT 3** instructions are stepped over to avoid a double breakpoint at the same address even though they appear as the next instruction to execute.

The following system routines are known to cause inconsistency or even system failure if traced. Consequently **Ptrace** will suspend tracing until after execution leaves these routines.

```
_Debug_CtrlC32      through _EndCtrlC32
_DebugLoadSymMTE    through EndDebugLoadSymMTE
_PGSwitchContext    through pgSwitchRet
```

See the [TX command](#) for information on tracing these routines.

**PN** suppresses the register display from the automatic [R command](#), but still displays an [unassembled](#) next instruction for each traced instruction. If the [ZS command](#) has been used to specify a different default command then **PN** behaves exactly as **P**.

An example of the output from **PN** is as follows:

```
##PN 5
0170:fff4521f 803d9e53e0ffff cmp     byte ptr [InterruptLevel (ffe0539e)],ff
0170:fff45226 75b4          jnz     fff451dc
0170:fff45228 803d9643e0ff00 cmp     byte ptr [_cTKNoBlock (ffe04396)],00
0170:fff4522f 75be          jnz     fff451ef
0170:fff45231 0f01e1        smsw    cx
##
```

**Note:** The last traced instruction is the next to be executed.

### **Warning:**

If any of the **PTrace** commands is interrupted, the Kernel Debugger may leave a temporary break-point active. This will result in a **Trap 1** when the system is next given control. If this occurs then either of the **PT** or **GT** commands will clear this condition.

-----

## Q - Quit the Dump Formatter



Quit the Dump Formatter.

### Syntax:

Q

**Parameters:**

*None*

**Results & Notes:**

The Dump Formatter is terminated.

**Warning:**

Do not use this command when the Dump Formatter is invoked from PMDF. This will cause PMDF to hang. To terminate the Dump Formatter either quit PMDF from the system menu or select another dump for processing.

---

## R - Set or Display Current CPU Registers



Display or set the current CPU registers saved on entry to the Kernel Debugger. Set default addresses for [E command](#), [D command](#), [K command](#) and [U command](#).

Under the Dump Formatter this command is implemented as an alias to the [.R command](#). Also applicable to the Dump Formatter only, the default addressing mode is not set according to the **VM** flags of the **EFLAGS** register but is assumed always to be in protect mode. This has been corrected from fix pack 29 of Warp 3.0 and base Warp 4.0.

The remaining discussion in the section applies to the Kernel Debugger.

**Syntax:**

R

T

*flag register*

*2-bit flag*

*16-bit register*

*32-bit register*

***flag register:***

F  
EF

flag mnemonics

CR0

cr0 flag mnemonics

MSW

msw flag mnemonics

**2-bit flag:**

IOPL                   pl

**16-bit register:**

AX  
BX  
CX                   16-bit value  
DX  
SI  
DI  
SP  
BP  
IP  
PC  
ES  
CS  
DS  
SS  
FS  
GS  
TR  
IDTL  
GDTL  
LDTR

**32-bit register:**

EAX  
EBX  
ECX                   32-bit value  
EDX  
ESI  
EDI  
ESP  
EBP  
EIP  
CR2  
CR3  
CR4  
DR0  
DR1  
DR2  
DR3  
DR6  
DR7  
TR6  
TR7  
IDTB  
GDTB

**Parameters:**

*(Default)*

Displays the current CPU registers on entry to the Kernel Debugger and sets default addresses for [E command](#), [D command](#) and [U command](#).

Register mnemonics are assigned the values displayed for use in address expressions and operands of other Kernel Debugger and Dump Formatter commands.

**Note:**

The [.SS command](#) may be used to change the displayed values of **CS**, **EIP**, **SS** and **ESP**. It does not affect the values restored then the Kernel Debugger returns control to the system.

**T**

Toggle register display mode between terse and non-terse forms. The terse form suppresses display of the test, debug, control, descriptor table and task registers.

This option affects both the **R** and **.R** commands.

### *flag register*

Specifies one of the flag registers to be modified. The following mnemonics may be used:

<b>F</b>	80286 <b>FLAGS</b> register.
<b>EF</b>	80486 <b>EFLAGS</b> register.
<b>MSW</b>	Machine status word.
<b>CR0</b>	Control register 0

Each of the flag bits is specified by a mnemonic. More than one flag may be specified, order being unimportant. The Kernel Debugger processes the flags from left to right, if an invalid flag is encountered processing stops, but those flags already processed remain in effect.

Some flags are toggled by specifying a single mnemonic, others use a one mnemonic for the set condition and a another of the reset condition.

If replacements flags are omitted then the user is prompted for values.

### *flag mnemonics*

Specifies one or more updated flags values for the **FLAGS** or **EFLAGS** registers.

The following mnemonics are defined. The value of *t* implies the flag value is toggled when the mnemonic is specified:

<i>Flag</i>	<i>Bit</i>	<i>Value</i>	<i>Description</i>
VM	17	t	Virtual 8086 Mode (EFLAGS only)
RF	16	t	Resume Flag - Disable Debug Exceptions (EFLAGS only)
NT	14	t	Nested Task
OV	11	1	Overflow
NV	11	0	¬Overflow
DN	10	1	Direction Down
UP	10	0	Direction Up
EI	9	1	Enable Interrupts
DI	9	0	Disable Interrupts
NG	7	1	Negative Sign
PL	7	0	Plus Sign
ZR	6	1	Zero Result
NZ	6	0	Non-zero Result
AC	4	1	Auxiliary Carry
NA	4	0	¬Auxiliary Carry
PE	2	1	Parity Even
PO	2	0	Parity Odd
CY	0	1	Carry
NC	0	0	¬Carry

### *cr0 flag mnemonics*

Specifies one or more updated flags values for the **CR0** register.

The following mnemonics are defined:

<i>Bit</i>	<i>Value</i>	<i>Flag</i>	<i>Description</i>
PG	31	1	Paging Enabled
ET	4	1	Extension Type Flag - x87 support
TS	3	1	Task Switch Flag
EM	2	1	Emulation exception
MP	1	1	Math Present
PM	0	1	Protect Mode Enabled

#### ***msw flag mnemonics***

Specifies one or more updated flags values for the **MSW** register.

The following mnemonics are defined:

<i>flag bit</i>	<i>value</i>	<i>description</i>
TS	3	1 Task Switch Flag
EM	2	1 Emulation exception
MP	1	1 Math Present
PM	0	1 Protect Mode Enabled

#### ***2-bit flag***

This option is used to specify that the **IOPL** field of the **FLAGS** or **EFLAGS** register should be updated with the specified replacement **2-bit value**. The mnemonic **IOPL** is coded to specify this option.

If the replacement value is not specified then the user is prompted for a value.

#### ***16-bit register***

This option is used to set the value of a register where *16-bit register* specifies either one of the standard **INTEL** register mnemonics or:

**GDTL** The **GDT** limit.

**IDTL** The **IDT** limit.

**PC** The program counter. This is synonymous with **IP**.

This option implies a request to update a register value. If the corresponding new **16-bit value** is not specified then the prompted for a replacement value.

#### ***32-bit register***

This option is used to set the value of a register where *32-bit register* specifies one of the standard **INTEL** register mnemonics or **GDTB** or **IDTB**.

This option implies a request to update a register value. If the corresponding new **32-bit value** is not specified then the prompted for a replacement value.

#### ***2-bit value***

Specifies the 2-bit replacement value for the **IOPL**.

#### ***16-bit value***

Specifies the 16-bit replacement value for a given 16-bit register.

#### ***32-bit value***

Specifies the 32-bit replacement value for a given 32-bit register.

#### **Results & Notes:**

The register information is obtained from a special save area when the Kernel Debugger is entered and restored from this area when control returns to the system.

When no operands are specified the **R** command operates in display mode in exactly the same manner as the [.R command](#).

From fix pack 29 for Warp 3.0 and base Warp 4.0 Pentium Processor support was added to the Kernel Debugger. This allows **CR4** to be specified as a register mnemonic, though CR4 is never displayed without specifying it explicitly as an operand to the **R** command. On non-Pentium systems, **CR4** is shown as **00000000**.

When operands are specified the **R** command. operates in alter mode. If no replacement value is supplied on the command then the user is prompted with the current value followed by a colon prompt character. For example:

```
##R SS
0030
:
```

Flag register value prompts have their current flag setting interpreted using the mnemonics described above. For example:

```
##R EF
--(rf) --(vm) --(nt) nv(ov) up(dn) ei(di) pl(ng) nz(zr) na(ac) po(pe) nc(cy)
:
```

This example shows mnemonics for current settings followed by their negating mnemonic in brackets. For example:

- RF** is not in effect, but since it is a toggle flag, the value **RF** specified at the prompt would set **RF**.
- NV** is in effect. To negate it, specify **OV** at the prompt.

## S - Search Memory for Data



Search a memory range for occurrences of a list of bytes.

**Syntax:**

```
S      addr      Ln      value
                        "string"
```

**Parameters:**

***addr*** The address of the memory location to be searched.

***Ln*** The number (*n*) of bytes to search.

***value*** A numerical byte value to be searched into memory. One or more values may be specified separated commas or blanks. These may be mixed with ***"string"*** values.

### *string*

A character string enclosed in single or double quotes. Each character is treated as a list byte values to search memory, no terminating 0x00 value is stored. No folding of characters to upper or lower case occurs. One or more strings may be specified separated by commas or blanks. These may be mixed with numerical *values*.

### Results & Notes:

The list of *values* and *strings* is used as a combined search argument. Only precise matches against the entire search argument are reported. The search is repeated for every byte location in the range specified. If no matches are found then nothing is displayed. Where matches are found the **Search** command displays a list of storage addresses. For example:

```
-----  
##s ptda_start 11000 "TD"  
0030:0000ffff  
ln 30:ffff  
0030:0000ffff os2krnl:TASKAREA:ptda_signature  
-----
```

If memory is present the storage is updated otherwise an **Invalid Address** message is generated. If this should happen, valid storage may be paged into memory by means of the [.I command](#).

## T - Trace Instruction Execution



Trace instruction execution singly or for a specific number or instructions or to a specific address.

### Syntax:

T				~
TX	=	start-addr	count	
TN				
TT				
TA			break-addr	~
TC	=	start-addr		
TS				

### Parameters:

(Default)

Trace one or more instructions, excluding known *bad* areas (see **X** subcommand below).

**A**

Trace all instructions to *break-addr*.

This option requires *break-addr* to be specified.

**C**

Counts all instructions executed until *break-addr* is reached.

**Note:** Counting is suspended when the system switches out of the current [context](#) in which the **TC** command was executed. It is resumed when that context switches back.

This option requires *break-addr* to be specified.

**N**

Trace instructions but suppress the register display after each instruction is executed.

**S**

The trace *special* option is similar to **TC** except that an intermediate instruction count is displayed before execution of each **CALL** instruction and after each return.



This option requires *break-addr* to be specified.

## Notes

Counting is suspended when the system switches out of the current [context](#) in which the **TS** command was executed. It is resumed when that context switches back.

**TS** does not attempt to match **CALL** with **RET** instructions. Instead it inserts a temporary [breakpoint](#) at the instruction address following the **CALL**. In addition **TS** maintains a 'stack' of return addresses and always checks the most recent two entries, as it single-instruction steps through the traced code, for a matching return address. This technique enables code that uses **JMP** instructions to return from a call to be better detected. *This is not a foolproof technique, especially where mutually recursive code is traced.*

## T

This option causes the Kernel Debugger's trap vector handlers to be removed temporarily from the [IDT](#) and the system's re-instated until after then next instruction has executed. After execution of the next instruction the the Kernel Debugger's [V commands](#) are re-instated.

This is a convenience option that saves manually unhooking a Kernel Debugger trap vector handlers from the [IDT](#). using a command sequence similar to:

```
VC n
T
VS n
```

## X

This option forces the Kernel Debugger to trace areas of system code that are known to be unsuitable for tracing. Normally, when **Trace** encounters one of the following routines:

```
_Debug_CtrlC32    through _EndCtrlC32
_DebugLoadSymMTE through EndDebugLoadSymMTE
_PGSwitchContext through pgSwitchRet
```

a temporary breakpoint is inserted at the routine's return address and the system is allowed to go to that address uninterrupted. When **TX** is used the Kernel Debugger will attempt to trace instructions within these routines.

*The consequence of forcing tracing in these routines may be at worst, the system is left in an unrecoverable state, and at best certain Kernel Debugger commands will give erroneous information.*

## *start-addr*

The address from which execution is to continue. This must be a valid address for the current context. If *start-addr* is omitted then execution continues from the current **CS:EIP**, as shown by the [R command](#).

**Warning:** Be very careful to ensure that the start address is valid for the privileged level and addressability of the code and data selectors in use. If the Kernel Debugger attempts to load a segment register that is invalid the system may trap in the debugger code.

## *break-addr*

The address at which tracing will stop and the Kernel Debugger will be re-entered unless one of the following conditions is encountered:

A fatal exception occurs

An Internal Processing Error (IPE) occurs.

A 'sticky' breakpoint fires

A non-maskable interrupt occurs

An **INT 3** instruction is executed

The user enters **Ctrl-C** from the debugging console. The **break-addr** only remains in effect until the Kernel Debugger is next re-entered.

## *count*

The number of instructions to trace before re-entering the Kernel Debugger unless one of the following conditions is encountered:

A fatal exception occurs

An Internal Processing Error (IPE) occurs.

A 'sticky' breakpoint fires

A non-maskable interrupt occurs

An **INT 3** instruction is executed

The user enters **Ctrl-C** from the debugging console.

If omitted then *count* defaults to **1** instruction.

### Results & Notes:

Except for **TN**, **TC** and **TS** the default command is executed when control returns to the debugging console. This defaults to the [R command](#) unless respecified through use of the [ZS command](#).

**TN** suppresses the register display from the automatic [R command](#), but still displays an [unassembled](#) next instruction for each traced instruction. If the [ZS command](#) has been used to specify a different default command then **TN** behaves exactly as **T**.

An example of the output from **TN** is as follows:

```
##TN 5
0170:fff4521f 803d9e53e0ffff cmp     byte ptr [InterruptLevel (ffe0539e)],ff
0170:fff45226 75b4          jnz     fff451dc
0170:fff45228 803d9643e0ff00 cmp     byte ptr [_cTKNoBlock (ffe04396)],00
0170:fff4522f 75be          jnz     fff451ef
0170:fff45231 0f01e1       smsw    cx
##
```

**Note:** The last traced instruction is the next to be executed.

**TC** displays the total number of instructions trace in the following message:

Total traced instructions: *nnnn* (decimal)

where *nnnn* is the number of traced instructions.

Following this message the default command is executed. See the [Z command](#) for details.

**TS** displays a variety of different messages, examples of which are:

```
-----
Instruction Count: 101
d0df:0000f319 9a0000c810      call     10c8:0000
```

Accumulated number of instructions executed before the **CALL** instruction.

```
-----
Exit: 108
```

Accumulated number of instructions executed when the return address is encountered.

**Note:** This does not include the instruction at the return address.

```
-----
...Special exit follows...
Exit: 360
```

Accumulated number of instructions executed when the second most recent return address is encountered. In this case the most recent return address is discarded from the 'stack'.

**Note:** This does not include the instruction at the return address.

```
-----
Switching context...
...Back in context
```

Signifies context switching occurring and the suspension and resumption of instruction counting.

-----  
Total traced instructions: nnnn (decimal)

The total number of instructions traced when the *break-addr* is encountered.

## Notes

**REP** and **REPNE** string instruction prefixes are handled differently to other instructions when single stepping. The Kernel Debugger generates a temporary break-point following the repeated string instructions (**MOVS**, **CMPS**, **SCAS**, **LDS** and **STOS**) and returns control to the system until the temporary break-point *fires*.

**INT 3** instructions encountered when single-stepping are reported but in actual fact stepped over, thereby avoiding a double break-point at the same address.

### Warning:

If any of the **Trace** commands is interrupted, the Kernel Debugger may leave a temporary break-point active. This will result in a **Trap 1** when the system is next given control. If this occurs then either of the **TT** or **GT** commands will clear this condition.

U - Unassemble



Unassemble storage at a given address.

### Syntax:

U

addr

### Parameters:

*addr*

The address of the storage location to be unassembled.

### Results & Notes:

The **U** command unassembles storage from the address given. No attempt is made to distinguish between code and data storage. If no *addr* is given then the default address is determined in order of precedence as follows:

- The last unassembled address + 1, or
- The **CS:EIP** of the last explicitly executed **.R command** or **R command**, or
- The address of the next instruction to be executed.

The **U** command takes its default addressing mode as set by the **R** or **.R** commands. Prior to fix pack 29 for Warp 3.0 and base Warp 4.0 V8086 addressing mode was ignored by the Dump Formatter unless explicitly specified by using the **&** addressing operator.

Output from the **U** command is in two forms depending on whether the storage address was set in the context of the default (Kernel

Debugger's or Dump Formatter's current) [thread slot](#) or another slot. In the former case output appears as in the following example:

```
-----
##u
0170:fff4521f 803d9e53e0ffff cmp    byte ptr [InterruptLevel (ffe0539e)],ff
0170:fff45226 75b4          jnz    fff451dc
0170:fff45228 803d9643e0ff00 cmp    byte ptr [_cTKNoBlock (ffe04396)],00
0170:fff4522f 75be          jnz    fff451ef
0170:fff45231 0f01e1        smsw   cx
0170:fff45234 66f7c10200    test   cx,0002
0170:fff45239 0f8552050000 jnz    fff45791
0170:fff4523f fa            cli
-----
```

In the latter case the context is shown by prefixing the [thread slot](#) to the address as in the following example:

```
-----
##.p*
Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
*0022# 0013 0003 0013 0001 blk 0300 7b6ea000 7b8c7128 7b8ab820 1eb8 18 epm
##.r 34

eax=00000000 ebx=000007f4 ecx=00000000 edx=0003ace7 esi=d02f4ef0 edi=000011ec
eip=0000272d esp=0000755e ebp=00007566 iopl=2 -- -- -- nv up ei ng nz na pe nc
cs=d02f  ss=001f  ds=bccf  es=ace7  fs=150b  gs=0000  cr2=15b20000  cr3=001d9000
doscall11:CONFORM16:postDOSSEMWAIT:
0034|d02f:0000272d c9          leave
##u

doscall11:CONFORM16:postDOSSEMWAIT:
0034|d02f:0000272d c9          leave
0034|d02f:0000272e ca0800    retf     0008
0034|d02f:00002731 87db      xchg    bx,bx
0034|d02f:00002733 90          nop
doscall11:CONFORM16:DOSSEMSET:
0034|d02f:00002734 c8040000  enter   0004,00
0034|d02f:00002738 8b4608    mov     ax,word ptr [bp+08]
0034|d02f:0000273b 3d0200    cmp     ax,0002
0034|d02f:0000273e 7448      jz      2788
0034|d02f:00002740 250300    and     a,0003
0034|d02f:00002743 3d0100    cmp     ax,0001
0034|d02f:00002746 7415      jz      275d
0034|d02f:00002748 8b4608    mov     ax,word ptr [bp+08]
##-----
```

**Note:**

The unassembled instruction mnemonics may be toggled between upper and lower case by use of the [Y command](#).

# V - Exception/Trap/Fault Vector Commands



This group of commands manipulates [IDT entries 0](#) through [e](#) to point to Kernel Debugger supplied interrupt handlers. By this means the Kernel Debugger may selectively be made to intercept each system exception before the system is allowed to process the exception. When a system exception is intercepted the Kernel Debugger gives control to the user. The original [IDT](#) entries are retained so that they may be re-instated, or given control following an exception which the Kernel Debugger has been intercepted. See the [GT](#) and [TT](#) commands for information in returning control to the system exception handlers.

**Syntax:**

V	L		
	S	R	interrupt
	T	V	
	C	P	,

F \*  
N  
U

## Parameters:

**L**

The **List** subcommand list active Kernel Debugger trap and interrupt [vectors](#).

Only a category specification (**R**, **V**, **P**, **F** or **N**) may be optionally specified with the **List** subcommand.

**S**

The **Set** subcommand activates a Kernel Debugger exception vector according to criteria specified in the remaining parameters. Vectors set using this option cause the Kernel Debugger to receive control only when the corresponding exceptions are generated in ring 2 and 3 code.

**T**

The **Trap** subcommand activates a Kernel Debugger exception vector according to criteria specified in the remaining parameters. Vectors set using this option cause the Kernel Debugger to receive control whenever the corresponding exceptions are generated regardless of the current privileged level.

**C**

The **Clear** subcommand re-instates one or more system exception handlers according to the criteria specified in the remaining parameters.

**R**

This option refines the exception criteria to **Real-mode** exceptions only.

If no refining category is specified then the vector subcommand being executed applies to the **R**, **V**, **P** and **F** options simultaneously.

**V**

This option refines the exception criteria to **V86-mode** exceptions only.

**P**

This option refines the exception criteria to **Protect-mode** exceptions only.

**F**

This option refines the exception criteria to those exceptions that would be 'fatal' to a process or the system. If a Local (system) Exception handler is registered then the exception is not intercepted. User Exception Handlers do not affect the operation of the Vector command. Local Exception Handlers protect the system from recoverable errors, in particular bad parameters passed in API calls. If a parameter causes the system to trap, the Local Exception Handler is given control and the application is terminated. VSF will not intercept such traps. For further information on exception handling and how to intercept exceptions in general, see [Trap and Exception Processing](#).

**U**

This option allows exceptions, fatal to a process, to be intercepted before the process is terminated. Interception occurs if the exception is not recovered by any user exception handler. **VSU** will intercept 'user-fatal' exceptions whether they originate from the system or the user. In particular if a local (system) exception handler has intercepted a kernel trap due to a bad API parameter then this will be intercepted if no user exception handler recovers from the error.

**Note:** This option was introduced with OS/2 Warp V3.0 fix pack 26 and OS/2 Warp V4.0 fix pack 1.

**N**

This option causes the Kernel Debugger exception handler to 'beep' continuously instead of giving control to the user. The user may then break into the Kernel Debugger by entering **Ctrl-C** at the debugging console.

The **N** option works in conjunction with the four refining categories, that is, it does not by itself cause an interrupt to be trapped but instead specifies an action when that event occurs.

The **N** option must be explicitly specified for all four subcommands (**L**, **S**, **T** and **C**) when required.

## *interrupt*

This allows one or more interrupt vectors, separated by commas, to be specified with the vector command as a refining criterion.

It is not valid with the **List** subcommand.

The abbreviation \* may be specified as an alternative to the following interrupts, in each of the refining categories:

Real-mode:	0,1,2,3,4,5,6
V86-mode:	0,1,3,4,5,6,7,9,a,b,c,d,e,(f,10)
Protect-mode:	0,1,3,4,5,6,7,9,a,b,c,d,e,(f,10)
Fatal option:	0,1,3,4,5,6,7,9,a,b,c,d,e,(f,10)
User-fatal option:	0,1,3,4,5,6,7,9,a,b,c,d,e,(f,10)
Noise option:	0,1,3,4,5,6,7,9,a,b,c,d,e,(f,10)

**Note:**

Vectors **f** and **10** are only allowed with OS/2 Warp SMP

**Results & Notes:**

Only the **List** subcommand gives immediate output, which is of the form in the following example:

```
-----
##VL
R 0 1 2 3 4 5 6
V
P d
U e
F e d
-----
```

As can be seen from this example each category is shown with its one-letter abbreviation followed by a list of interrupt vectors currently being intercepted by the Kernel Debugger

**Notes:**

The **N** option must be specified explicitly to be listed.

The **U** option is only from OS/2 Warp V3.0 fix pack 26 and OS/2 Warp V4.0 fix pack 1.

All other subcommands only cause output to appear when an interrupt is intercepted. When this happens the following events occur:

1. The **N** option is checked, if enabled the Kernel Debugger emits a continuous beep until the user breaks in through the debugging console.
2. A trap message is issued if the [default command](#) is set to the [R command](#).
3. The default command is executed.

The following figure shows the format of the trap messages issued by the Kernel Debugger exception handlers:

```
Trap 0 - Divide Error Exception
Trap 1 - Unexpected trace interrupt
Trap 2 - NMI Interrupt
Trap 4 - INTO Detected Overflow Exception
Trap 5 - BOUND Range Exceeded Exception
Trap 6 - Invalid Opcode Exception
Trap 7 - Processor Extension Not Available Exception
Trap 8 - Double Exception Detected nnnn
Trap 9 - Processor Extension Segment Overrun
Trap 10 (0AH) - Invalid TSS nnnn, xxxxxxxx
Trap 11 (0BH) - Segment Not Present nnnn, xxxxxxxx
Trap 12 (0CH) - Stack Segment Overrun or Not Present nnnn, xxxxxxxx
Trap 13 (0DH) - General Protection Fault nnnn, xxxxxxxx
Trap 14 (0EH) - Page Fault nnnn, xxxxxxxx
```

In the messages above **nnnn** is substituted with the Intel exception code and **xxxxxxxx** is substituted with an interpretation of the Intel error code flags. For **Trap 10**, **Trap 11**, **Trap 12** and **Trap 13** the error code flags are interpreted as:

<b>External</b>	External event
-----------------	----------------

<b>IDT Gate</b>	IDT gate selector error
<b>GDT</b>	GDT selector error
<b>LDT</b>	LDT Selector error

For **Trap 14** the error code flags are interpreted as a combination of:

<b>Not Present</b>	Page not present
<b>Read Access</b>	Read Access failure
<b>Write Access</b>	Write Access Failure
<b>User mode</b>	Fault occurred when executing in User mode
<b>Supervisor</b>	Fault occurred when executing in Supervisor mode

If a trap occurs in the debugger component of the Kernel Debugger the trap message will be appended with:

- In Debugger

If this happens then the only hope of recovering the system is to set the registers, using the [R command](#), to a known consistent set of values.

See the **INTEL x86 Programmer's Reference** or the **INTEL Pentium User's Guide** for further information on exceptions and error codes.

#### Notes

**Trap 1** normally occurs as part of the operation of the Kernel Debugger. Only unexpected **Trap 1** exception are therefore reported.

When a **Trap 1** is generated through use of the Debug Registers then the Kernel Debugger signals this with the message **Debug register hit**.

**Trap 3** occurs through use of the **INT 3** instruction. This is used both by the Kernel Debugger and user programs in implementing [break-points](#). User programs may use the **INT 3** instruction as a program controlled technique for breaking into the debugger. In these cases a trap message is not displayed.

-----

## W - Withmap Add/Remove



Add or remove a symbol map. Under the Kernel Debugger this merely activates or deactivates a symbol map. Under the Dump Formatter a symbol file may be re-loaded using the **Withmap** command.

#### Syntax:

```
WA      ~
WR      map-name
        symbol-file
        *
```

#### Parameters:

**A**                      Activate 1 or all symbol maps.

**Note:** If the corresponding load module is not active then the map will remain deactivated. See the [L command](#) for more information on displaying map status.

<b>R</b>	Remove 1 or all symbol maps.
<b>L</b> (not shown)	This subcommand applied only to the Dump Formatter and has been superseded by the <a href="#">.LM command</a> .
<b><i>map-name</i></b>	The symbol map name to be activated or deactivated
<b><i>symbol-file</i></b>	The symbol file name, with optional path and extension, to be loaded or removed.
<b>Note:</b>	
	This operand applied only to the Dump Formatter
<b>*</b>	Specifies all maps or symbol files should be loaded or removed.

**Results & Notes:**

An error message is displayed only if the specified **map-name** is not loaded.

See the [L command](#) for related information.

## Y - Set or Display Dump Formatter and Kernel Debugger Options



Set or display Dump Formatter and Kernel Debugger disassembly and register options.

**Syntax:**

<b>Y?</b>	~
<b>Y</b>	~
	386ENV
	REGTERSE
	DISLWR

**Parameters:**

<i>(Default)</i>	Display current option settings.
<b>?</b>	Display help for the <b>Y</b> command.
<b>386ENV</b>	Force the Kernel Debugger and Dump Formatter to toggle the environment setting between <b>286</b> and <b>386</b> modes.  This affects the way in which commands interpret the register set. For example, in <b>286</b> mode, general registers are assumed, by default to be 16-bit registers. Under rare circumstances it necessary to force a particular mode to obtain a correct disassembly listing from the <a href="#">U command</a> . Mostly this occurs in system code that is multi-modal and has juxtaposed sections of 32-bit and 16-bit code.  The initial setting is 386 mode under OS/2 V2.0 and above.
<b>REGTERSE</b>	This has the same effect as the <a href="#">RT command</a> .



The initial setting is for terse register display.

#### DISLWR

This option toggles upper and lower case display of assembler mnemonics from the [U command](#).

The initial setting is for lower case mnemonics.

#### Results & Notes:

No information is displayed when setting options.

When querying options those in effect are displayed, for example:

```
-----  
##y  
386env dislwr regterse  
-----
```

This shows that 386 environment is assumed, lower case disassembly is in effect and terse register display is active. If any one of these setting is toggled then the corresponding flag is not displayed.

## Z - Set, Execute or Display the Default Command



Set, execute or display the default command.

#### Syntax:

```
z [L | S] [cmd]
```

#### Parameters:

*(Default)*

Execute the default command string.

**L**

Display the default command string

**S**

Set the default command string.

*cmd*

Specifies a Dump Formatter or Kernel Debugger commands to be use in the default command string. If the command string comprises more than one command the each must be separated by commas and the entire string enclosed in quotes.

#### Results & Notes:

The default command string is executed automatically at breakpoints (where no other command string is associated with the breakpoint), after instruction tracing or when exception vectors are trapped. See the following commands for more information:

[B commands](#)

[G command](#)

[P command](#)

[T commands](#)

[V commands](#)

When the Kernel Debugger and Dump Formatter are initialised the default command string is set to "R".

**Note:**

When the user breaks into the Kernel Debugger with **Ctrl-C** the [R command](#) is executed regardless of the default command setting.

-----

## External Commands

The following comprise the set of external commands:

<a href="#">.?</a>	Display external command help
<a href="#">.A</a>	Display the SAS structure
<a href="#">.B</a>	Set COM Parameters
<a href="#">.C</a>	Display the Common ABIOS Data Area
<a href="#">.D</a>	Display an OS/2 System Structure
<a href="#">.H</a>	Display Dump Data Set Inforamtion
<a href="#">.I (KDB)</a>	Swap in Storage
<a href="#">.I (DF)</a>	Display Dump State
<a href="#">.K</a>	Display Ring 3 stack
<a href="#">.LM</a>	Format Loader structures (MTE, OTE, STE)
<a href="#">.M</a>	Formate Memory Structures
<a href="#">.MA</a>	Format Memory Arena records (VMAR)
<a href="#">.MC</a>	Format Memory Context Records (VMCO)
<a href="#">.MK</a>	Format Memory Lock Information Records (VMLI)
<a href="#">.ML</a>	Format Memory Alias Records (VMAL)
<a href="#">.MO</a>	Format Memory Object Records (VMOB)
<a href="#">.MP</a>	Format Memory Physical Page Frame Tables
<a href="#">.MV</a>	Format Memory Virtual Frame Tables
<a href="#">.N</a>	Display Dump Header Information
<a href="#">.O</a>	Override default behaviour
<a href="#">.P</a>	Display Process and Thread Status Information
<a href="#">.PB</a>	Display Blocked Process Information
<a href="#">.PQ</a>	Display Scheduler Thread Queuing Information
<a href="#">.PU</a>	Display Process and Thread User Space Information
<a href="#">.R</a>	Display ring 2/3 registers

<a href="#">.REBOOT</a>	Reboot the system under test
<a href="#">.S</a>	Switch default thread slot
<a href="#">.SYSDUMP</a>	Force a System Dump and Restart the System
<a href="#">.T</a>	Format the System Trace Buffer

-----

## .? - Show External Command Help



Display help for internal Kernel Debugger and Dump Formatter commands.

### Syntax:

. ?

**Parameters:** *None.*

### Results & Notes:

Displays a help summary for most of the Dump Formatter and Kernel Debugger external commands.

### **Note:**

Prior to fix pack 29 for Warp V3, some of the help information displayed is out-of-date. This information has been updated from fix pack 29 of Warp V3 and base Warp V4.

Two pages of information are displayed with an intervening **--More--** prompt.

-----

## .A - Format the System Anchor Segment (SAS)



Format the System Anchor Segment ([SAS](#)).

### Syntax:

. A

**Parameters:**

*none*

### Results & Notes:

The SAS is located from either GTD selector 70 or 78.

**.A** displays the following information:

```

--- SAS Base Section ---
        SAS signature: SAS
        offset to tables section: 0016
        FLAT selector for kernel data: 0168
        offset to configuration section: 001E
        offset to device driver section: 0020
        offset to Virtual Memory section: 002C
        offset to Tasking section: 005C
        offset to RAS section: 006E
        offset to File System section: 0074
        offset to infoseg section: 0080
--- SAS Protected Modes Tables Section ---
        selector for GDT: 0008
        selector for LDT: 0000
        selector for IDT: 0018
        selector for GDTPOOL: 0100
--- SAS Device Driver Section ---
        offset for the first bimodal dd: 0CB9
        offset for the first real mode dd: 0000
        sel for Drive Parameter Block: 04C8
        sel for BIOS prot. mode CDA: 0000
        seg for BIOS real mode CDA: 0000
        selector for FSC: 00C8
--- SAS Task Section ---
        selector for current PTDA: 0030
        FLAT offset for process tree head: FFF10910
        FLAT address for TCB address array: FFF06BB6
        offset for current TCB number: FFDFFB5E
        offset for ThreadCount: FFDFFB62
--- SAS File System Section ---
        handle to MFT PTree: FE72CFBC
        selector for System File Table: 00C0
        sel. for Volume Parameter Bloc: 0788
        sel. for Current Directory Struc: 07B8
        selector for buffer segment: 00A8
--- SAS Information Segment Section ---
        selector for global info seg: 0428
        address of curtask local infoseg: 03C80000
        address of DOS task's infoseg: FFFFFFFF
        selector for Codepage Data: 07CB
--- SAS RAS Section ---
        selector for System Trace Data Area: 04B0
        segment for System Trace Data Area: 04B0
        offset for trace event mask: 0B28
--- SAS Configuration Section ---
        offset for Device Config. Table: 0D50
--- SAS Virtual Memory Mgt. Section ---
        Flat offset of arena records: FFF13304
        Flat offset of object records: FFF1331C
        Flat offset of context records: FFF1330C
        Flat offset of kernel mte records: FFF0A891
        Flat offset of linked mte list: FFF07934
        Flat offset of page frame table: FFF11A70
        Flat offset of page range table: FFF111EC
        Flat offset of swap frame array: FFF03BAC
        Flat offset of Idle Head: FFF10090
        Flat offset of Free Head: FFF10080
        Flat offset of Heap Array: FFF11B78
        Flat offset of all mte records: FFF12E04

```

Each of the items displayed has the following significance:

#### --- SAS Base Section ---

Marks the beginning of the **SAS** header section.

#### SAS signature

**SAS** signature from **SAS\_signature** (SAS+0x0). Always set to character value "SAS ".

#### offset to tables section

Offset from SAS selector to the protected mode tables section.

Taken from **SAS\_tables\_data** (SAS+0x4).

**FLAT selector for kernel data**

Selector for 4G Read/Write addressability.

Taken from **SAS\_flat\_sel** (SAS+0x6).

**offset to configuration section**

Offset from SAS selector to the configuration tables section.

Taken from **SAS\_config\_data** (SAS+0x8).

**offset to device driver section**

Offset from SAS selector to the device driver section.

Taken from **SAS\_dd\_data** (SAS+0xa).

**offset to Virtual Memory section**

Offset from SAS selector to the Virtual Memory section.

Taken from **SAS\_vm\_data** (SAS+0xc).

**offset to Tasking section**

Offset from SAS selector to the Tasking section.

Taken from **SAS\_task\_data** (SAS+0xe).

**offset to RAS section**

Offset from SAS selector to the RAS data section.

Taken from **SAS\_RAS\_data** (SAS+0x10).

**offset to File System section**

Offset from SAS selector to the File System section.

Taken from **SAS\_file\_data** (SAS+0x12).

**offset to infoseg section**

Offset from SAS selector to the Infoseg section.

Taken from **SAS\_info\_data** (SAS+0x1e).

**--- SAS Protected Modes Tables Section ---**

Marks the beginning of the protected mode tables section

**selector for GDT**

GDT selector that maps the GDT.

Taken from **SAS\_tbl\_GDT** (SAS\_tables\_section+0x0).

**selector for LDT**

No longer used.

Taken from **SAS\_tbl\_LDT** (SAS\_tables\_section+0x2).

**selector for IDT**

GDT selector that maps the IDT

Taken from **SAS\_tbl\_IDT** (SAS\_tables\_section+0x4).

**selector for GDTPOOL**

First GTD selector in selector pool. I.e first non-predefined GDT selector.

Taken from **SAS\_tbl\_GDTPOOL** (SAS\_tables\_section+0x6).

**--- SAS Device Driver Section ---**

Marks the beginning of the Device Driver Section

**offset for the first bimodal dd**

Offset from SAS selector to the first device driver header in the device driver chain.

See [.D command](#) for formatting device driver headers.

Taken from **SAS\_dd\_bimodal\_chain** (SAS\_dd\_section+0x0).

**offset for the first real mode dd**

No longer used.

Taken from **SAS\_dd\_real\_chain** (SAS\_dd\_section+0x2).

**sel for Drive Parameter Block**

Selector that points to the head of the [DPB](#) chain.

See [.D command](#) for formatting **DPBs**.

Taken from **SAS\_dd\_DPB\_segment** (SAS\_dd\_section+0x4).

**sel for BIOS prot. mode CDA**

Selector for BIOS protect mode CDA.

See [.C command](#) for displaying CDA information.

Taken from **SAS\_dd\_CDA\_anchor\_p** (SAS\_dd\_section+0x6).

**seg for BIOS real mode CDA**

Segment for BIOS real mode CDA. See [.C command](#) for displaying CDA information.

Taken from **SAS\_dd\_CDA\_anchor\_r** (SAS\_dd\_section+0x8).

**selector for FSC**

Selector for the [FSC segment](#).

Taken from **SAS\_dd\_FSC** (SAS\_dd\_section+0x2).

**--- SAS Task Section --**

Marks the beginning of the tasking section

**selector for current PTDA**

Selector for the current PTDA and ring 0 stack.

Taken from **SAS\_task\_PTDA** (SAS\_task\_section+0x0).

**FLAT offset for process tree head**

Linear address of **\_pPTDAFirst**, which contains the linear address of the [PTDA](#) that heads the PTDA tree.

Taken from **SAS\_task\_ptdaptrs** (SAS\_task\_section+0x2).

**FLAT address for TCB address array**

Linear address of **\_papTCBSlots**, which contains the linear address of the [TCB](#) array.

Taken from **SAS\_task\_threadptrs** (SAS\_task\_section+0x6).

**offset for current TCB number**

Linear address of **\_TaskNumber**, which contains the current [thread slot number](#).

Taken from **SAS\_task\_tasknumber** (SAS\_task\_section+0xa).

**offset for ThreadCount**

Linear address of **\_ThreadCount**, which contains the highest thread slot number in use - 1.

Taken from **SAS\_task\_threadcount** (SAS\_task\_section+0xe).

**--- SAS File System Section --**

Marks the beginning of the File System Section

**handle to MFT PTree**

Linear address of the head of the [Ptree](#) for the .

See [.D command](#) for formatting **MPTs**.

Taken from **SAS\_file\_MFT** (SAS\_file\_section+0x0).

**selector for System File Table**

Selector for the segment containing a table of selectors that point to tables of [SFTs](#). Each SFT table contains an 8 byte header followed by contiguous SFT entries.

See [.D command](#) for formatting **SFTs**.

Taken from **SAS\_file\_SFT** (SAS\_file\_section+0x4).

**sel. for Volume Parameter Bloc**

This is the selector for the work buffer used by volume mount processing.

Taken from **SAS\_file\_VPB** (SAS\_file\_section+0x6).

**Note:**

The selector for the **VPB** segment is not given by this field. It may be located from the selector named by global variable **GDT\_VPB** See **.D command** for formatting **VPBs**.

**sel. for Current Directory Struc**

Selector for the **RMP** segment containing **CDS** structures.

See **.D command** for formatting **CDSs**.

Taken from **SAS\_file\_CDS** (SAS\_file\_section+0x8).

**selector for buffer segment**

Selector for the file system buffer segment.

Taken from **SAS\_file\_buffers** (SAS\_file\_section+0xa).

**--- SAS Information Segment Section --**

Marks the beginning of the Information Section.

**selector for global info seg**

Selector for the Global Information Segment (**GISEG**).

Taken from **SAS\_info\_global** (SAS\_info\_section+0x0).

**address of curtask local infoseg**

16:16 far pointer for the current Local Information Segment (**LISEG**).

Taken from **SAS\_info\_global** (SAS\_info\_section+0x2).

**address of DOS task's infoseg**

Real mode local information segment pointer (unused).

Taken from **SAS\_info\_localRM** (SAS\_info\_section+0x6).

**selector for Codepage Data**

Selector for the segment containing the **Code Page Data Information Block** (CDIB).

Taken from **SAS\_info\_CDIB** (SAS\_info\_section+0xa).

**--- SAS RAS Section --**

Marks the beginning of the RAS section

**selector for System Trace Data Area**

Selector for the **STDA** trace buffer.

Taken from **SAS\_RAS\_STDA\_p**

See **.T command** for formatting the system trace buffer. (SAS\_RAS\_section+0x0).

**segment for System Trace Data Area**

Selector for the **STDA** trace buffer.

Taken from **SAS\_RAS\_STDA\_r** (SAS\_RAS\_section+0x2).

The same value is stored in both **SAS\_RAS\_STDA\_p** and **SAS\_RAS\_STDA\_r**.

**offset for trace event mask**

Offset from the **SAS** to the trace major event codes table (**ras\_mec\_table**).

Taken from **SAS\_RAS\_event\_mask** (SAS\_RAS\_section+0x4).

**--- SAS Configuration Section --**

Marks the beginning of the Configuration section

**offset for Device Config. Table**

Offset from the **SAS** to the device configuration table.

Taken from **SAS\_config\_table** (SAS\_config\_section+0x0).

### --- SAS Virtual Memory Mgt. Section ---

Marks the beginning of the Virtual Memory Management section

#### Flat offset of arena records

The linear address of **\_parvmOne**, the linear address of the first VM arena record (**VMAR**).

See **.MA command** for related information.

Taken from **SAS\_vm\_arena** (SAS\_vm\_section+0x0).

#### Flat offset of object records

The linear address of **\_pobvmOne**, the linear address of the first VM object record (**VMOB**).

See **.MO command** for related information.

Taken from **SAS\_vm\_object** (SAS\_vm\_section+0x4).

#### Flat offset of context records

The linear address of **\_pcovmOne**, the linear address of the first VM context record (**VMCO**).

See **.MC command** for related information.

Taken from **SAS\_vm\_context** (SAS\_vm\_section+0x8).

#### Flat offset of kernel mte records

The linear address of **\_DosModMTE**, the kernel (**DOSCALLS.DLL**) **MTE**.

See **.LM command** for related information.

Taken from **SAS\_vm\_krnl\_mte** (SAS\_vm\_section+0xc).

#### Flat offset of linked mte list

The linear address of **\_global\_h**, the linear address of the head of the **MTE** chain of link library modules.

See **.LM command** for related information.

Taken from **SAS\_vm\_glbl\_mte** (SAS\_vm\_section+0x10).

#### Flat offset of page frame table

The linear address of **\_pft**, the linear address of the first (frame 0) page frame structure (**PF**).

See **.MP command** for related information.

Taken from **SAS\_vm\_pft** (SAS\_vm\_section+0x14).

#### Flat offset of page range table

The linear address of **\_pgPageablePAI**, the pageable **PAI**. The first double word of the PAI points to the page range table.

Taken from **SAS\_vm\_prt** (SAS\_vm\_section+0x18).

#### Flat offset of swap frame array

The linear address of **\_smbmDF**, the linear address of swap frame allocation bit map followed by **\_smFileSize**, the swap file size word length value in pages.

Taken from **SAS\_vm\_swap** (SAS\_vm\_section+0x1c).

#### Flat offset of Idle Head

The linear address of **\_pgIdleList**, which points to the pseudo-PF at the head of the idle **PF** list.

See **Idle Page Frame Structures** for more information locating Idle Page Frame Structures.

See **.MP command** for related information.

Taken from **SAS\_vm\_idle\_head** (SAS\_vm\_section+0x20).

#### Flat offset of Free Head

The linear address of **\_pgFreeList**, which points to the pseudo-PF at the head of the free **PF** list.



See [Free Page Frame Structures](#) for more information locating Free Page Frame Structures.

See [.MP command](#) for related information.

Taken from **SAS\_vm\_free\_head** (SAS\_vm\_section+0x24).

#### Flat offset of Heap Array

The linear address of **\_apkh**, the array of [VMKH](#) kernel heap header structures. Note: the first entry is unused.

Taken from **SAS\_vm\_heap\_info** (SAS\_vm\_section+0x28).

#### Flat offset of all mte records

The linear address of **\_mte\_h**, which is the linear address of the head of the [MTE](#) chain.

See [.LM command](#) for related information.

Taken from **SAS\_vm\_all\_mte** (SAS\_vm\_section+0x2c).

-----

## .B - Select the Communications Port and Speed



Select the communications port and speed.

#### Syntax:

```
.B          speed          port
```

#### Parameters:

##### *speed*

The **COMx** port speed. Any of the following values are valid:

150t  
300t  
600t  
1200t  
2400t  
4800t  
9600t  
19200t

**Note:** Since baud rates are usually expressed in decimal and the default number base for the Kernel Debugger is hexadecimal then a **t** subscript must be supplied when using decimal values.

##### *port*

Specifies which **COM** port is to be used. If **1** or **2** are specified then **COM1** or **COM2** are implied. Any other numeric value is assumed to be an I/O port address.

#### Results & Notes:

When the Kernel Debugger initialises a default baud rate of **9600t** is set.

The **COM** port defaults to **COM2** if there are two serial ports otherwise **COM1** unless no comports are defined in the ROM BIOS data area,

in which case the first port address in the ROM BIOS data area is assumed.

The parity, data and stop bit settings default to none, 8 and 1. These may be altered either:

from the Kernel Debugger by writing directly to the **COM** port control register using the **O command**  
or from the system under test by using the **MODE** command.

If synchronisation is lost with the debugging console, for example because the debugging communications port has been temporarily used by another application then it may be reset using the **MODE** command. from the command line of the system under test. For example, to re-specify the default parameters use:

```
MODE COM2 9600,n,8,1
```

## .C - Display the Common ABIOS Data Area



Display ABOIS Command Data Area information.

**Syntax:**

```
.C
```

**Parameters:** None

**Results & Notes:**

.C displays data for each logical device ID anchored from the [Common ABIOS Data Area \(CDA\)](#). If the ABIOS is not present or initialised then the following message is displayed:

```
ABIOS Not Present or Not Initialised
```

The presence of ABIOS is indicated by a non-zero byte value located at symbol:

**ABIOS\_Present.**

If the ABIOS is present and initialised then data based on the Logical Device ID (LID) table is displayed. The LID Table is located from a selector located at:

**ABIOS\_CDS\_ANCHOR\_p** - in protect mode, or

**ABIOS\_CDS\_ANCHOR\_r** - in real mode

Tabular data of the following form is displayed:

LID(0000)	Type=Reserved	DB=001e:0114	FTT=0000:0000
LID(0001)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(0002)	Type=Internal	DB=0438:06f0	FTT=0448:011c
LID(0003)	Type=Diskette	DB=0438:0728	FTT=0448:012c
LID(0004)	Type=Video	DB=0438:07a4	FTT=0448:017c
LID(0005)	Type=Keyboard	DB=0438:07e4	FTT=0448:01e4
LID(0006)	Type=Printer	DB=0438:080c	FTT=0448:0238
LID(0007)	Type=Asynch	DB=0438:082c	FTT=0448:0280
LID(0008)	Type=SysTimer	DB=0438:084c	FTT=0448:02e8
LID(0009)	Type=RTCTimer	DB=0438:0860	FTT=0448:0328
LID(000a)	Type=SysService	DB=0438:087c	FTT=0448:0380
LID(000b)	Type=NMInterrupt	DB=0438:08a0	FTT=0448:03cc
LID(000c)	Type=PointDevice	DB=0438:08d8	FTT=0448:0404
LID(000d)	Type=DMA	DB=0438:08f0	FTT=0448:044c
LID(000e)	Type=Security	DB=0438:0920	FTT=0448:04a4
LID(000f)	Type=POS	DB=0438:0938	FTT=0448:04f0

LID(0010)	Type=CMOSRam	DB=0438:0960	FTT=0448:0538
LID(0011)	Type=ErrorLog	DB=0438:0978	FTT=0448:0574
LID(0012)	Type==	DB=0438:0990	FTT=0448:05ac
LID(0013)	Type=Disk	DB=0438:09d8	FTT=0448:060c
LID(0014)	Type=anonymous)	DB=0438:0a50	FTT=0448:0684
LID(0015)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(0016)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(0017)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(0018)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(0019)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(001a)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(001b)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(001c)	Type=NULL	DB=0000:0000	FTT=0000:0000
LID(001d)	Type=NULL	DB=0000:0000	FTT=0000:0000

**Note:** There is a formatting error that is illustrated in **LID 12** and **LID 14** lines. See description below of **type=** parameter for an explanation of this!

The fields displayed have the following meaning:

LID

Logical Device ID.

This is a sequential numbering of entries that appear in the table of LID entries. The entry, LID(0000), is however a dummy entry mapped by CDAType where the selector:offset of DB= are number of LID entries and offset to table of data pointers. Data pointer entries have one of the following forms:

Field Name	Off	Len	Type	Description
DataPtr	+0	6		Data Pointer in CDA
DLimit	+0	2	W	Limit Field
DOffset	+2	2	W	Offset Field
DSegment	+4	2	W	Segment Field

Field Name	Off	Len	Type	Description
PhysPtr	+0	6		Physical Data Pointer (INTEL Format)
	+0	2	W	Limit Field
PhysLSW	+2	2	W	Lo Order 16 bits
PhysMSW	+4	2	W	Hi Order 16 bits

Type=

This an interpretation of the device type (**Devid**) field taken from the corresponding Device Block. The following Type values may appear:

Reserved	Used only for the LID(0000) dummy entry.
Null	signifies an unused entry (DB=0000:0000).
Internal	Devid=0000 used for internal ABIO calls.
Diskette	Devid=0001 Diskette device.
Disk	Devid=0002 Disk device.
Video	Devid=0003 Video device.
Keyboard	Devid=0004 Keyboard.
Printer	Devid=0005 Printer.
Asynch	Devid=0006 Asynchronous device.
SysTimer	Devid=0007 System Timer.
RTCTimer	Devid=0008 RTC Timer.
SysService	Devid=0009 SysService.

NMIInterrupt	Devid=000a NMI Interrupt.
PointDevice	Devid=000b Pointer Device.
LightPen	Devid=000c Light Pen.
JoyStick	Devid=000d JoyStick.
CMOSRam	Devid=000e CMOS RAM.
DMA	Devid=000f DMA controller.
POS	Devid=0010 Programmable Option Select.
ErrorLog	Devid=0011 Error Log.
S/A Dump	Devid=0012 Stand Alone Dump.
IOPortAlloc.	Devid=0013 I/O Port Allocation.
Audiotone	Devid=0014 Audio device.
Int/8259	Devid=0015 Interrupt Controller.
Security	Devid=0016 Keyboard Security.

Other Device Types are in use but are not translated to a predictable name.

For example:

Devid=0017	SCSI Subsystem Interface
Devid=0018	SCSI Peripheral

Where this occurs the Devid may be found at offset +8 of the device block.

DB=sel:off

sel:off address of the Device Block for the corresponding LID. The device block has the following standard header structure:

<i>Field Name</i>	<i>Off</i>	<i>Len</i>	<i>Type</i>	<i>Description</i>
DeviceBlock	+0	8		Device Block Header
DevBlength	+0	2	W	Device Block Length
Revision	+2	1	B	Revision
	+3	1	B	Reserved
	+4	2	W	Logical ID
Devid	+6	2	W	Device ID

FTT=sel:off

sel:off address to the Function Transfer Table for this LID. The FTT has the following standard header structure:

<i>Field Name</i>	<i>Off</i>	<i>Len</i>	<i>Type</i>	<i>Description</i>
FTTTable	+0	16		Function Transfer Table Header
FStart	+0	4	D	Start Routine Entry Point
FInt	+4	4	D	Interrupt Routine Entry Point
FTimeO	+8	4	D	Start Routine Entry Point
FuncCount	+c	2	W	Count of Functions
	+e	2	W	Reserved

-----

## .D - Display an OS/2 System Structure



Display an OS/2 System Structure.

**Syntax:**

.D	SFT	addr	~
	VPB		
	DPB		
	CDS		
	KSEM		
	DT		
	DEV		
	REQ		
	MFT		
	BUF		
	BPB		
	SEM32		
	MUXQ		
	OPENQ		

**Parameters:**

**structure**

The structure type may take one of the following values:

<b>SFT</b>	Format a file system <a href="#">System File Table</a> entry.
<b>VPB</b>	Format a file system <a href="#">Volume Parameter Block</a> .
<b>DPB</b>	Format a file system <a href="#">Drive Parameter Block</a> .
<b>CDS</b>	Format a file system <a href="#">Current Directory Structure</a> .
<b>KSEM</b>	Format a <a href="#">Kernel Semaphore</a> .
<b>DT</b>	Disk Trace is now obsolete.
<b>DEV</b>	Format a device driver header.
<b>REQ</b>	Format a device driver request packet.
<b>MFT</b>	Format a <a href="#">Master File Table</a> entry.
<b>BUF</b>	Format a file system I/O buffer.
<b>BPB</b>	Format a <a href="#">BIOS Parameter Block</a> .
<b>SEM32</b>	Format a 32-bit semaphore.
<b>MUXQ</b>	Format a <b>mutex</b> semaphore wait queue.
<b>OPENQ</b>	Format a 32-bit semaphore open queue.

**addr**

Specifies the address of the structure to be formatted. If omitted then the current **DS** selector value, offset **0** is assumed.

An [address expression](#) may be specified.

**Results & Notes:**

**Warning:**

.D will format OS/2 structures without any validation. It is entirely incumbent on the user to ensure that the address used does in fact point to the named structure. Failure to observe this caution will result in meaningless information being displayed.

The following are examples of each of the 13 formatted structures. Refer to the [System Reference](#) for a description of each formatted structure.

<a href="#">SFT</a>	System File Table Entry
<a href="#">VPB</a>	Volume Parameter Block
<a href="#">DPB</a>	Drive Parameter Block
<a href="#">CDS</a>	Current Directory Structure

KSEM	Kernel Semaphore
DEV	Device Driver Header
REQ	Device Driver (Strategy 1) Request Packet
MFT	Master File Table Entry
BUF	File System Buffer
BPB	BIOS Parameter Block
SEM32	32-Bit Semaphore
MUXQ	Semaphore MUX Queue
OPENQ	Semaphore Open Queue

**Note:**

The Dump Formatter prior to fix pack 29 for Warp V3 did not format structures contained in segments less than 512 bytes, typically .D DEV and .D DPB. This is fixed from fix pack 29 of Warp V3 and base Warp V4.

## System File Table Entry (SFT)

```
.d sft d0:8
    sf_ref_count: 0001
    sf_usercnt: 0000
    reserved: 00
    sf_flags(2): 0100:0000
    sf_devptr: #0000:0000
    sf_FSC: #00c8:0008
    sf_chain: #0000:0000
    sf_MFT: fe7fb788
sfdFAT_firFILEclus: 5ad6
sfdFAT_cluspos: 09c8
sfdFAT_lstclus: 0000
sfdFAT_dirsec: 00000000
sfdFAT_dirpos: 00
sfdFAT_name:
sfdFAT_EAHandle: 0000
    sf_plock: 0000
    sf_NmPipeSfn: 0000
    sf_codepage: 0000
    sfi_mode: 00a0
    sfi_hVPB: 0012
    sfi_ctime: 0000
    sfi_cdate: 0000
    sfi_atime: 0000
    sfi_adate: 0000
    sfi_mtime: 0000
    sfi_mdate: 0000
    sfi_size: 000bb135
    sfi_position: 00085d90
    sfi_UID: 0000
    sfi_PID: 0000
    sfi_PDB: 0000
    sfi_selfsfn: 0000
    sfi_tstamp: 00
    sfi_DOSattr: 20
##
```

System File Table Entry

**Notes:**

The **sfdFAT\_name** is only meaningful for SFTs that represent open [FAT](#) file system files.

For a description of the SFT fields see the [System File Table Entry \(SFT\)](#) in the **System Reference**.

## Volume Parameter Block (VPB)

```
##ln gdt_vpb
0138:00000098 os2krnl:DOSGDTDATA:GDT_VPB
##.d vpb 98:12
    vpb_flink: 0000
    vpb_blink: 008d
    vpb_ref_count: 007b
    vpdFAT_cluster_mask: 41
    vpdFAT_cluster_shift: 00
    vpdFAT_first_FAT: 00b2
```

```

vpb_search_count: 0000
vpb_first_access: 00
vpb_signature: 444a
vpb_flags(2): 02:00
vpb_FSC: #00c8:0008
vpi_ID: 26715015
vpi_pDPB: #04b8:00c4
vpi_cbSector: 0200
vpi_totsec: 0007cfe0
vpi_trksec: 0020
vpi_nhead: 0040
vpi_pDCS: #0000:0000
vpi_pVCS: #0000:0000
vpi_drive: 07
vpi_unit: 07
vpi_text: UNLABELED
vpi_flags: 0003

vpdFAT_FAT_count: a8
vpdFAT_root_entries: 0004
vpdFAT_first_sector: 885c0400
vpdFAT_max_cluster: 410e
vpdFAT_FAT_size: b200
vpdFAT_dir_sector: aa04a800
vpdFAT_media: 0d
vpdFAT_next_free: 00b2
vpdFAT_free_cnt: 04a8
vpdFAT_FATentrysize: b2
vpdFAT_IDsector: 00000000
vpdFAT_access: 0000
vpdFAT_accwait: 0000
vpdFAT_pEASFT: #0000:0000

####.m 98:0

*har      par      cpg      va      flg next prev link hash hob      hal
0003 %feaeef04c 00000400 %fe6ef000 001 0002 0023 0000 0000 0003 0000      =0000
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0003 0003 fec5 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
pvmli    cs      eip      phlock cpg      va      flg hptda hob sig csig
%fe82e4c4 002d 0a6800a5 %ac22403c 0001 %fe83c000 0005 024b 0003 ea9f ea9f
##dd %(98:0)-10 l8
%fe6fd4dc 00000001 ff5905b8 5e02bd64 000009bd
%fe6fd4ec 05d6007b 000009ae 0000099c dac40000
##dd %(98:0)-10-4+9bc l8
%fe6fde94 6d6b6d62 00180000 ffa20098 00e800e8
%fe6fdea4 ffc2001c 00000008 00000000 00000000
##.mo ffa2
ffa2 vpb
##

```

## Volume Parameter Block

### Notes:

The selector for the **VPB** segment may be found by listing the symbol **GDT\_VPB** and using its offset.

The handle of a **VPB** (**hVPB**) is the offset within the **VPB** segment.

The VPB segment has a unique owner ID, which may be determined using the [.M command](#).. In the case of the VPB segment it is allocated from the kernel resident heap, so the true owner id is found in the heap header (or its extension - the trailer).

For a description of the VPB fields see the [Volume Parameter Block \(VPB\)](#) in the **System Reference**.

## Drive Parameter Block (DPB)

```

.d dpb 4b8:c4
    dpb_drive: 07
    dpb_unit: 07
    dpb_driver_addr: #0798:0000
    dpb_next_dpb: #04b8:00e0
    dpb_cbSector: 0200
    dpb_first_FAT: 0001
    dpb_toggle_time: 00000000
    dpb_hVPB: 0012
    dpb_media: f8
    dpb_flags: 20
    dpb_drive_lock: 0000
    dpb_strategy2: #07a0:139c

```

```
##.m 04b8:0c4
```

```

*har      par      cpg      va      flg next prev link hash hob      hal
0003 %feaeef04c 00000400 %fe6ef000 001 0002 0023 0000 0000 0003 0000      =0000
hob      har hobjxt flgs own  hmte  sown,cnt lt st xf
0003 0003 fec5 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
      pvml  cs      eip      phlock  cpg      va      flg hptda hob sig  csig
%fe82e4c4 002d 0a6800a5 %ac22403c 0001 %fe83c000 0005 024b 0003 ea9f ea9f
##dd %(4b8:0) - 10 18
%fe6f1638 00000000 ff360498 2000fe6f 000001c5
%fe6f1648 00000000 001c0798 020004b8 00000001
##dd %(4b8:0) - 10-4+1c4 18
%fe6f17f8 00000000 00000000 ff9604b8 0000000a
%fe6f1808 ff46000c 000014f8 00d020c8 ff46000c
##.mo ff96
ff96 dpb

```

## Drive Parameter Block

### Notes:

The **DPB** may be located from the **VPB**.

The DPB segment has a unique owner ID, which may be determined using the [.M command](#). In the case of the VPB segment it is allocated from the kernel resident heap, so the true owner Id is found in the heap header (or its extension - the trailer).

For a description of the DPB fields see the [Driver Parameter Block \(DPB\)](#) in the **System Reference**.

## Current Directory Structure (CDS)

```

>> locate the SAS file system section
##dw 70:12 11
0070:00000012 0074
##dw 70:74
0070:00000074 0fa4 fe70 00c0 07f8 0828 00a8 0428 0000
0070:00000084 03c8 ffff ffff 0843 0000 0000 0000 0000
0070:00000094 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000a4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000b4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000c4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000d4 0000 0000 0000 0000 0000 0000 0000 0000
0070:000000e4 0000 0000 0000 0000 0000 0000 0000 0000

>> +8 into the file system section is the CDS RMP selector.
>> Can verify this by checking out the memory object owner.

##.m 828:0

*har      par      cpg      va      flg next prev link hash hob      hal
0003 %feaeef04c 00000400 %fe6ef000 001 0002 0023 0000 0000 0003 0000      =0000
hob      har hobjxt flgs own  hmte  sown,cnt lt st xf
0003 0003 fec5 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
      pvml  cs      eip      phlock  cpg      va      flg hptda hob sig  csig
%fe82e4c4 002d 0a6800a5 %ac22403c 0001 %fe83c000 0005 024b 0003 ea9f ea9f

>> Owned by the Kernel Resident Heap. Look at the header

##dd %(828:0)-10 18
%fe7015b4 000007d0 ff5c07b0 0000bd64 0000060d
%fe7015c4 049d0600 01ae0014 00000001 00000400

>> This is an attributed block so look at the trailer

##dd %(828:0)-10-4+60c 18
%fe701bbc 00000000 00000000 ff610828 0000fe70
%fe701bcc ff9e0054 4d45534b 00000201 00000000

##.mo ff61
ff61 cdsrmp

```



>> 828 does indeed point to the CDS RMP

>> Now dump the CDS handle table for the process of interest

```
##.p#
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
0048# 0029 0004 0029 0001 blk 0200 ab805000 ab99b820 ab97fc20 led4 11 cmd
```

```
##dw %ab99b820 +cds_handle-ptda_start 11a
%ab99bac8 0000 0000 0000 0000 0000 0000 0000 0090
%ab99bad8 0000 0000 0000 0000 0000 0000 0000 0000
%ab99bae8 0000 0000 0000 0000 0000 0000 0000 0000
%ab99baf8 0000 0000
```

>> Except for driver 07 (H:) the current directory handle is null.  
>> This implies that the current directory for drive H: is not the  
>> root. To see which it is, we need to locate the the CDS entry with  
>> handle 0x0090.

>> The RMP has a 0x14 byte header. Each entry is prefixed with a word  
>> length followed by the handle for that entry.

>> Starting with the first entry scan through until handle 0x0090 is  
>> located.

```
##dw 828:14 14
0828:00000014 8028 0000 00b2 0000
```

```
##dw 828:14 12
0828:00000014 8028 0000
```

```
##dw 828:14+28 12
0828:0000003c 0028 001c
```

```
##dw 828:14+28+28 12
0828:00000064 0028 001d
```

```
##dw 828:14+28+28+28 12
0828:0000008c 0026 001e
```

```
##dw 828:14+28+28+28+26 12
0828:000000b2 8023 0014
```

```
##dw 828:14+28+28+28+26+23 12
0828:000000d5 0028 0022
```

```
##dw 828:14+28+28+28+26+23+28 12
0828:000000fd 002e 008f
```

```
##dw 828:14+28+28+28+26+23+28+2e 12
0828:0000012b 0025 0090
```

>> The CDS starts after the length prefix.

```
##.d cds 828:12b+2
cd_handle: 0090                      cddFAT_id: 0000
cd_refcnt: 0002
cd_flags: 40
cd_devptr: 04b8:00c4
cd_OwnerFSC: 0008

cdi_hVPB: 0012
cdi_end: 0002
cdi_flags: 80
cdi_text: H:\spool
##
```

Current Directory Structure

#### Notes:

The selector for the **CDS** segment may be located from the SAS, as illustrated above, or from the storage at label **CDSAddr**.

The CDS RMP has a unique owner ID, which may be determined using the [.M command](#). In the case of the CDS RMP, it is allocated from the kernel resident heap, so the true owner id is found in the heap header (or its extension - the trailer).

For a description of the CDS fields see the [Current Directory Structure \(CDS\)](#) in the **System Reference**.

## Kernel Semaphore (KSEM)

```
>> Intra-Process serialisation mutex KSEM imbedded in the PTDA.
>> This KSEM is sometimes referred to as "fscrit" (file system critical
>> section) since it is used to serialise file system activity within
>> a process.
.p#
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
*000c# 0002 0000 0002 0004 blk 0804 ab78d000 ab997020 ab978420 1c9c 00 cntrl
##.d ksem %ab997020 +ptda_ptdasem-ptda_start
Signature      : KSEM                      Nest: 0000
Type           : MUTEX
Flags          : 00
Owner          : 0000                      PendingWriters: 0000
##

>> MFT Shared KSEM imbedded at the beginning of the MFT
##..d sft d0:8
sf_ref_count: 0001                      sfi_mode: 00a0
sf_usercnt: 0000                      sfi_hVPB: 0012
reserved: 00                          sfi_ctime: 0000
sf_flags(2): 0100:0000                sfi_cdate: 0000
sf_devptr: #0000:0000                 sfi_atime: 0000
sf_FSC: #00c8:0008                   sfi_adate: 0000
sf_chain: #0000:0000                  sfi_mtime: 0000
sf_MFT: fe7fb788                      sfi_mdate: 0000
sfdFAT_firFILEclus: 5ad6                sfi_size: 000bb135
sfdFAT_cluspos: 09c8                    sfi_position: 00085d90

##.d ksem %fe7fb788
Signature      : KSEM                      Nest: 0000
Type           : SHARE                    Readers: 0000
Flags          : 01                      PendingReaders: 0000
Owner          : 0000                    PendingWriters: 0000
##

>> Slot 49 is blocked. So we proceed by finding out what the BlockId
>> represents by finding its owner.

##.pb 49
Slot Sta BlockID Name Type Addr Symbol
0049 blk fe83bdf4 warp_d

##.m %0fe83bdf4
*har par cpg va flg next prev link hash hob hal
072c %feaf8dd2 00000400 %00540000 149 072d 072b 0003 0000 0003 0025 hptda=0878
hal=0025 pal=%ffe5d140 har=072c hptda=0878 pgoff=00000 f=021
har par cpg va flg next prev link hash hob hal
0003 %feaeef04c 00000400 %fe6ef000 001 0002 0023 0000 0000 0003 0000 =0000
hob har hobnxt flgs own hmte sown,cnt lt st xf
0003 072c fec5 1000 ffec 0000 0000 00 01 00 00 vmkrhrw
pvmlt cs eip phlock cpg va flg hptda hob sig csig
%fe82e4c4 002d 0a6800a5 %ac22403c 0001 %fe83c000 0005 024b 0003 ea9f ea9f

>> Block Id is in the kernel resident heap - assume that it is at the
>> beginning of a data portion of a heap block.
>> Dump the header.

##dd %0fe83bdf4-10 l8
%fe83bde4 00000000 bd100000 fe83fe83 ff7e0018
```

```
%fe83bdf4 4d45534b 000a0004 46380001 bd28a801

>> Object Owner Id is ff7e
##.mo ff7e
ff7e ksem
##.d ksem %0fe83bdf4
Signature      : KSEM
Type           : EVENT
Flags          : 04
Owner          : 000a                PendingWriters: 0001
##
```

## Kernel Semaphore

### Notes:

KSEMs are usually found imbedded in system control blocks for serialisation and sharing purposes.

Dynamically allocated KSEMs are allocated out of one of the kernel heaps.

Virtual Device Driver semaphore helper services result in KSEMs.

Under the ALLSTRICT kernel only, the KSEM has a signature field. This is manufactured by the .D command for non-ALLSTRICT kernels. Under the ALLSTRICT kernel the presence of a KSEM may be verified by dumping the KSEM in bytes. Offset **+0x0** is where the signature is located.

The owner field refers to the [slot number](#) of the semaphore owner.

When a thread blocks on a KSEM the following addresses are used as the **BlockId**

MUTEX KSEM	Address of the beginning of the KSEM <a href="#">structure</a> .
SHARED KSEM	Address of the Pending Readers count field within the KSEM <a href="#">structure</a> .
EXCLUSIVE KSEM	Address of the Pending Writers count field within the KSEM <a href="#">structure</a> .

To format KSEM from a BlockId, locate the beginning of the KSEM, either by dumping a few bytes before the BlockId address. The signature will be visible under the ALLSTRICT kernel, which is at the beginning of the KSEM. If the KSEM is allocated from one of the Kernel's heaps then the KSEM object id, **0xff7e** will occur in the heap header which prefixes the beginning of the KSEM.

For a description of the KSEM structure see the [Kernel Semaphore Structure](#) in the **System Reference**.

-----

## Physical Device Driver Header (DEV)

>> Driver header address taken from the VBP with handle 12:

```
.d vpb 98:12
    vpb_flink: 0000                vpdFAT_cluster_mask: 41
    vpb_blink: 008d                vpdFAT_cluster_shift: 00
    vpb_ref_count: 007a            vpdFAT_first_FAT: 00b2
    vpb_search_count: 0000         vpdFAT_FAT_count: a8
    vpb_first_access: 00           vpdFAT_root_entries: 0004
    vpb_signature: 444a            vpdFAT_first_sector: 885c0400
    vpb_flags(2): 02:00            vpdFAT_max_cluster: 410e
    vpb_FSC: #00c8:0008            vpdFAT_FAT_size: b200
    vpi_ID: 26715015               vpdFAT_dir_sector: aa04a800
    vpi_pDPB: #04b8:00c4           vpdFAT_media: 0d
    vpi_cbSector: 0200             vpdFAT_next_free: 00b2
    vpi_totsec: 0007cfe0           vpdFAT_free_cnt: 04a8
    vpi_trksec: 0020               vpdFAT_FATentrysize: b2
    vpi_nhead: 0040                vpdFAT_IDsector: 00000000
    vpi_pDCS: #0000:0000           vpdFAT_access: 0000
    vpi_pVCS: #0000:0000           vpdFAT_accwait: 0000
```

```

        vpi_drive: 07                                vpdFAT_pEASFT: #0000:0000
        vpi_unit: 07
        vpi_text: UNLABELED
        vpi_flags: 0003
##.d dpb 4b8:c4
        dpb_drive: 07
        dpb_unit: 07
        dpb_driver_addr: #0798:0000
        dpb_next_dpb: #04b8:00e0
        dpb_cbSector: 0200
        dpb_first_FAT: 0001
        dpb_toggle_time: 00000000
        dpb_hVPB: 0012
        dpb_media: f8
        dpb_flags: 20
        dpb_drive_lock: 0000
        dpb_strategy2: #07a0:139c

##.d dev 798:0
        DevNext: 0778:0000
        DevAttr: 2880
        DevStrat: 0dbc
        DevInt: 0000
        NumUnits: 0c
        DevProtCS: 07a0
        DevProtDS: 0798
        DevRealCS: 0000
        DevRealDS: 0000

```

>> Formatting the device driver directly from the 1st data segment.

```

##.lmo 'ibmkbd
hnte=009b pnte=%fe6f1fb8 mflags=8008e1c9 h:\ibmkbd.sys
seg  sect psiz vsiz hob  sel  flags
0001 0001 0194 01fa 0000 0540 8d49 data iter prel rel
0002 0002 11c2 11c4 0000 0548 8d60 code shr prel rel
##.d dev 540:0
        DevNext: 0510:0000
        DevAttr: 8980
        DevStrat: 0680
        DevInt: 0586
        DevName: IBMKBD$
        DevProtCS: 0548
        DevProtDS: 0540
        DevRealCS: 0000
        DevRealDS: 0000
##

```

## Physical Device Driver Header

### Notes:

The Device Header appears in one of two formats, depending on whether the device supports multiple units or not.

**DevInt** is not the interrupt routine offset, as it was for DOS device drivers. Under OS/2 this is the offset to the Inter-Device Driver Communications (IDC) Entry Point.

For a description of the DEV fields see the [Physical Device Driver Header \(DEV\)](#) in the **System Reference**.

---

## Device Driver (Strategy 1) Request Packet (REQ)

>> The two request packet pools for general device driver use:

```
##.moc 93
```

```
*har      par      cpgr      va      flg next prev link hash hob      hal
0091 %feaeffc80 00000010 %ab546000 129 0090 0092 0000 0000 0093 0000      sel=04a8
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0093 0091 0000 0124 ff40 0000 0000 00 00 00 00 reqpkt1
##.moc 94
```

```
*har      par      cpgr      va      flg next prev link hash hob      hal
0092 %feaeffc96 00000010 %ab536000 129 0091 0093 0000 0000 0094 0000      sel=04b0
hob      har hobnxt flgs own  hmte  sown,cnt lt st xf
0094 0092 0000 0124 ff33 0000 0000 00 00 00 00 reqpkt2
```

>> Formatting the request packet assigned to slot 43:

```
##.p#
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
*0043# 000d 0004 000d 0005 blk 081f ab7fb000 ab99ac20 ab97f220 1d24 17 faxworks

##dd %ab97f220 +lac 11
%ab97f3cc 04a80832

##.d req 4a8:832
      PktLen: 2c
      PktUnit: 00
      PktCmd: 04
      PktStatus: 0000
      PktDOSLink: 00000000
      PktDevLink: 00000000
      IOmedia: 00
      IOpData: 008f7883
      IOcount: 0000
      IOstart: 00000000
```

>> Formatting the request packet assigned to slot 2:

```
##.p2
Slot Pid Ppid Csid Ord Sta Pri pTSD pPTDA pTCB Disp SG Name
0002 0001 0000 0000 0002 blk 0200 ab779000 ffe3ca00 ab977020 1f3c 00 *tsd
##dd %ab977020 +lac 11
%ab9771cc 04a80012
##.d req 4a8:12
      PktLen: 32
      PktUnit: 00
      PktCmd: 00
      PktStatus: 0000
      PktDOSLink: 00000000
      PktDevLink: 00000000
      InitcUnit: 00
      InitDevHlp: 0000:0000
      InitParms: 0000:0000
      InitDrv: 00
      InitSysiData: 0000
##
```

## Device Driver Request Packets

### Notes:

Request Packets are allocated from one of three pools:

Strategy 1 request pool

Strategy 2 request pool

Swapper request pool

Each thread is pre-assigned a strategy 1 request packet. If this is in use when a device driver tries to allocate another then a packet is allocated from the strategy 2 pool for strategy 1 use.

Asynchronous Read and Write requests are implemented in DOSCALL1.DLL by creating multiple threads on which to run the parallel I/O requests.

.D REQ does not format Strategy 2 format Request Packets.

For a description of the Request Packet fields see the [Device Driver Request Packer](#) in the **System Reference**.

## Master File Table Entry (MFT)

>> Display the SFT for SFN 20

```
##.d sft d0:(83*20+8)
    sf_ref_count: 0001                sfi_mode: 00c2
    sf_usercnt: 0000                 sfi_hVPB: 0000
    reserved: 00                     sfi_ctime: 0000
    sf_flags(2): 00c0:0000           sfi_cdate: 0000
    sf_devptr: #0af0:0000            sfi_atime: 0000
    sf_FSC: #00c8:ff40              sfi_adate: 0000
    sf_chain: #00d0:170f             sfi_mtime: b19d
    sf_MFT: fe82ff7c                sfi_mdate: 1f5f
sfdFAT_firFILEclus: 0000            sfi_size: 00000000
    sfdFAT_cluspos: 0000             sfi_position: 00000000
    sfdFAT_l
```

>> From the SFT display the MFT

```
.d mft %fe82ff7c
    mft_ksem:
Signature      : KSEM                Nest: 0000
Type           : SHARE               Readers: 0000
Flags          : 01                  PendingReaders: 0000
Owner          : 0000                PendingWriters: 0000
    mft_lptr: 0000                   mft_sptr: 00d0:08bb
    mft_pCMap: 00000000              mft_serl: 013e      mft_signature: 466d
    mft_CMapKSem:
    mft_hvpb: 0000                  mft_opflags: 0000      mft_flags: 0000
    mft_name: \DEV\MOUSE$
```

>> Display the SFT for SFN 40

```
##.d sft d0:(83*40+8)
    sf_ref_count: 0001                sfi_mode: 1302
    sf_usercnt: 0000                 sfi_hVPB: 008d
    reserved: 00                     sfi_ctime: 0000
    sf_flags(2): 0000:0000           sfi_cdate: 0000
    sf_devptr: #0000:0000            sfi_atime: 0000
    sf_FSC: #00c8:0008              sfi_adate: 0000
    sf_chain: #00d0:214b             sfi_mtime: 0000
    sf_MFT: fe6f190c                sfi_mdate: 0000
sfdFAT_firFILEclus: 470c            sfi_size: 00000000
    sfdFAT_cluspos: 09c8             sfi_position: 00000000
    sfdFAT_l
```

>> From the SFT display the MFT

```
.d mft %fe6f190c
    mft_ksem:
Signature      : KSEM                Nest: 0000
Type           : SHARE               Readers: 0000
Flags          : 01                  PendingReaders: 0000
Owner          : 0000                PendingWriters: 0000
    mft_lptr: 0000                   mft_sptr: 00d0:2045
    mft_pCMap: 00000000              mft_serl: 00a3      mft_signature: 466d
    mft_CMapKSem:
    mft_hvpb: 008d                  mft_opflags: 0000      mft_flags: 0001
    mft_name: D:\SWAPPER.DAT
##
```

Master File Table Entries

## Notes:

The MFT entry may be located from each SFT that represents an open instance of a file.

The MFT points to the most recent SFT open instance of the file.

The Dump Formatter prior to fix pack 29 for Warp V3 did not format the MFT correctly under the ALLSTRICT kernel. This is fixed from fix pack 29 of Warp V3 and base Warp V4.

For a description of the MFT field, see the [Master File Table Entry \(SFT\)](#) in the **System Reference**.

## File System Buffer (BUF)

>> Locate the file system buffer segment

```
##ln gdt_buffers
0138:000000a8 os2krnl:DOSGDTDATA:GDT_Buffers
```

```
##dw a8:0
00a8:00000000 ade4 94c4 0000 ff93 005a 0218 bc8b 0000
00a8:00000010 0664 0200 bc8c 000f 9000 00e5 0234 ade4
00a8:00000020 0000 0279 e05e 0000 0001 0400 0000 0000
00a8:00000030 0000 0000 46e5 4e49 3030 3630 4d54 2050
00a8:00000040 0000 0000 0000 0000 0000 b5cd 1f5e 3251
00a8:00000050 0400 0000 46e5 4e49 3030 3730 4d54 2050
00a8:00000060 0000 0000 0000 0000 0000 b5c8 1f5e 0000
00a8:00000070 0000 0000 4de5 3046 3030 2038 4d54 2050
```

```
##.d buf a8:ade4
  buf_next: 001c          buf_prev: ffff          buf_freeLink: 0000
  buf_flags: 02          buf_hVPB: 0279          buf_sector: 00000001
  buf_tid: 00           buf_wrtcnt: 02          buf_wrtcntinc: 0096
  buf_fill: 0000

##.d buf a8:234
  buf_next: 044c          buf_prev: 001c          buf_freeLink: 0000
  buf_flags: 04          buf_hVPB: 0279          buf_sector: 0000e05d
  buf_tid: 00           buf_wrtcnt: 01          buf_wrtcntinc: 0000
  buf_fill: 0000

##.d buf a8:44c
  buf_next: 0664          buf_prev: 0234          buf_freeLink: 0000
  buf_flags: 02          buf_hVPB: 0279          buf_sector: 0000001c
  buf_tid: 00           buf_wrtcnt: 02          buf_wrtcntinc: 0096
  buf_fill: 0000

##.d buf a8:664
  buf_next: 087c          buf_prev: 044c          buf_freeLink: 0000
  buf_flags: 04          buf_hVPB: 0279          buf_sector: 0000bcd4
  buf_tid: 00           buf_wrtcnt: 01          buf_wrtcntinc: 0000
  buf_fill: 0000
```

>> Find the volume these buffers are assigned to.

```
##ln gdt_vpb
0138:00000098 os2krnl:DOSGDTDATA:GDT_VPB
```

```
##.d vpb 98:279
  vpb_flink: 01fe          vpdFAT_cluster_mask: 07
  vpb_blink: 02f4          vpdFAT_cluster_shift: 03
  vpb_ref_count: 004e      vpdFAT_first_FAT: 0001
  vpb_search_count: 0000   vpdFAT_FAT_count: 02
  vpb_first_access: 09     vpdFAT_root_entries: 0200
  vpb_signature: 444a      vpdFAT_first_sector: 0000014d
  vpb_flags(2): 02:40      vpdFAT_max_cluster: 95f5
  vpb_FSC: #0000:ff40      vpdFAT_FAT_size: 0096
  vpi_ID: e2ea4414         vpdFAT_dir_sector: 0000012d
  vpi_pDPB: #04b8:0038    vpdFAT_media: f8
  vpi_cbSector: 0200      vpdFAT_next_free: 0000
```

```

vpi_totsec: 0004b0f0          vpdFAT_free_cnt: 63a8
vpi_trksec: 003f             vpdFAT_FATentrysize: 10
vpi_nhead: 0010             vpdFAT_IDsector: 00000000
vpi_pDCS: #0000:0000        vpdFAT_access: 0000
vpi_pVCS: #0000:0000        vpdFAT_accwait: 0000
vpi_drive: 02               vpdFAT_pEASFT: #00d0:23da
vpi_unit: 02
vpi_text: DOS FAT
vpi_flags: 0003

```

```

>> The file system buffer segment is assigned a unique object owner
>> id.

```

```

##.m 0a8:0
*har      par      cpg      va      flg next prev link hash hob      hal
0003 %feaef04c 00000400 %fe6ef000 001 0002 0023 0000 0000 0003 0000      =0000
hob      har hobnxt flgs own hmte sown,cnt lt st xf
0003 0003 fec5 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
pvmli     cs      eip      phlock  cpg      va      flg hptda hob sig  csig
%fe82e4c4 002d 0a6800a5 %ac22403c 0001 %fe83c000 0005 024b 0003 ea9f ea9f
##dd %(a8:0)-10 18
%fe702ff0 0b0c0001 4d5000a2 fe705854 0000cc99
%fe703000 94c4ade4 ff930000 0218005a 0000bc8b
##dd %(a8:0)-10-4+cc98 18
%fe70fc84 00000000 00000000 ff9300a8 003ec010
%fe70fc94 ffa4000c fe80caac 00020100 ff9e0014
##.mo ff93
ff93 fsbuf
##

```

## File System Buffer (BUF)

### Notes:

File system buffers are allocated out of a buffer segment whose selector may be located either from the [SAS File System](#) section, offset **+0xa** or from symbol **GDT\_BUFFERS**.

The buffer segment contains a header of length **+0x1c**.

Header Offset **+0x0** gives the offset to the head of the list of most recently used buffers.

Header Offset **+0x4** gives the offset to the tail of the list of most recently used buffers.

Each buffer contains a **0x18** byte header followed by **0x200** bytes of data. The buffer header is what is formatted by **.D BUF**.

For a description of the Buffer Header fields, see the [File System Buffers](#) in the **System Reference**.

-----

## BIOS Parameter Block (BPB)

```

##.d bpb bootbp
  BytesPerSector: 0200
SectorsPerCluster: 08
  ReservedSectors: 0001
    NumberOfFATs: 00
    RootEntries: 0200
    TotalSectors: 0000
  MediaDescriptor: f8
    SectorsPerFAT: 00fa
    SectorsPerTrack: 0020
      Heads: 0040
    HiddenSectors: 00000820
    BigTotalSectors: 0007cfe0

##.d bpb minimumbpb
  BytesPerSector: 0200

```



SectorsPerCluster: 01  
ReservedSectors: 0001  
NumberOfFATs: 00  
RootEntries: 0010  
TotalSectors: 0000  
MediaDescriptor: f0  
SectorsPerFAT: 0001  
SectorsPerTrack: 0009  
Heads: 0001  
HiddenSectors: 00000000  
BigTotalSectors: 003fffff

BIOS Parameter Block (BPB)

Notes:

- Two system BPBs are locatable at symbols **BootBPB** and **minimumBPB**.
- Others are pointed to from the Device Driver Request Packet for **DosDevIOCtl** command code 2 (build **BPB**).
- See [.D REQ Command](#) for information on formatting Device Driver Request Packets.

For a description of the BPB fields, see the [BIOS Parameter Block](#) in the **System Reference**.

32-Bit Semaphore Structures (SEM32, OPENQ and MUXQ)

```
##.pb 25
Slot Sta BlockID Name Type Addr Symbol
0025 blk fe81d2d0 pmsHELL Sem32 8001 004b hevLazyWrite

##.d sem32 %fe81d2d0
Type: Shared Event
Flags: Reset
pMuxQ: 00000000
Post Count: 0000
pOpenQ: fe56eb10
pName: fd074e98
Create Addr: 13f60088

##.pb 2f
Slot Sta BlockID Name Type Addr Symbol
002f blk fe86ffdc pmsHELL

##.d sem32 %fe86ffdc
Type: Shared Event
Flags: Reset
pMuxQ: 00000000
Post Count: 0000
pOpenQ: fe56eb02
pName: NULL (anonymous)
Create Addr: 12d16b48

##.pb 30
Slot Sta BlockID Name Type Addr Symbol
0030 blk fe86fe58 pmsHELL Sem32 0001 00ce hevSleeper

##.d sem32 %fe86fe58
Type: Private Event
Flags: Reset
pMuxQ: 00000000
Post Count: 0000
Open Count: 0001
Create Addr: 13f62f28
```

### Three types of Event Semaphore

#### Notes:

32-bit semaphores may be Event or Mutex in type, private or shared in scope and if shared, named or anonymous.

The BlockId of a thread waiting on a 32-bit semaphore is the address of the semaphore structure. The **Type** field in the [.PB command](#) usually indicates a 32-bit semaphore when in use, however this is not always the case. The next example shows how to determine precisely whether the blockid points to a 32-bit semaphore.

```
##.pb 33
Slot Sta BlockID Name Type Addr Symbol
0033 blk fe86falc pmsHELL
##.m %0fe86falc

*har par cpg va flg next prev link hash hob hal
0003 %feaef04c 00000400 %fe6ef000 001 0002 0023 0000 0000 0003 0000 =0000
hob har hobnxt flgs own hmte sown,cnt lt st xf
0003 0003 fec5 0000 ffec 0000 0000 00 01 00 00 vmkrhrw
pvmlt cs eip phlock cpg va flg hptda hob sig csig
%fe82e380 002d 0a6800a5 %ac22403c 0001 %fe83c000 0005 024b 0003 ea9f ea9f
##dd %0fe86falc-10 l8
%fe86fa0c 12d15b4c 54564553 ab97d220 ffc20018
%fe86falc 00000010 00000000 c4280001 455000a7
##.mo ffc2
ffc2 semstruc
##.d sem32 %0fe86falc
Type: Private Event
Flags: Reset
pMuxQ: 00000000
Post Count: 0000
Open Count: 0001
Create Addr: 00a7c428

##
```

How to determine whether a BlockId points to a 32-bit semaphore.

#### Notes:

Except for RAMSEM, MUXWAIT, ChildWait and private conventions the BlockId is an address of a structure or routine that relates to the resource or event being waited for.

The [.M command](#) is used to identify the owner of the BlockId. In this case it is the kernel resident heap.

Each resident heap block is prefixed with a 4 byte header. If the low order bit is 0 then the high word of the header contains the owner of the heap block.

32-bit Semaphore structures are allocated from regular resident heap blocks. Thus the owner id may be seen by displaying storage before the BlockId address.

```
##.pb 56
0056 blk fe88ad8c mutxwait Sem32 8001 0090 _WINOS2_Settings + 77

##.d sem32 %fe88ad8c
Type: Shared Mutex
Flags:
pMuxQ: fe88ab94
Request Ct: 0001
Owner: 0055
Requester Ct: 0001
pOpenQ: fe5724c2
pName: fd084368
Create Addr: 00022e98

##.d openq %fe5724c2

PID Open Count
```

```

-----
00d8      0001
00d7      0001
00d6      0001

##da %fd084368
%fd084368 RJM\MUTEX0

##.d muxq %fe88ab94

pMux
-----
fe88aba4

##.d sem32 %fe88aba4
      Type: Shared MuxWait
      Flags: Mutex_Mux
      SR Count: 0003
      SR Pointer: fe88a8f4
      Wait Count: 0001
      pOpenQ: fe571e94
      pName: fd0843c8
      Create Addr: 00022ec4

##da %fd0843c8
%fd0843c8 RJM\MUXWAIT
##.d openq %fe571e94

PID      Open Count
-----
00d7      0001
00d6      0001

##dd %fe88a8f4
%fe88a8f4  00000003 00000004 80000090 00000000
%fe88a904  80000091 00000001 80000094 00000001
%fe88a914  ffbe0014 fe88aba4 00000000 5158554d
%fe88a924  fe88a916 ffc70010 fe88a958 dd2f4580
%fe88a934  000001a2 ffc70010 fe88aa88 dd2f464d
%fe88a944  00000010 ffa4000c fe88a968 000102d5
%fe88a954  ffc70010 fe88a9d8 1bd49ce0 000101a5
%fe88a964  ffa4000c fe88aa7c 0001038d ffc70010

##

```

Mux Wait Semaphores

#### Notes:

**pOpenQ** points to an Open Queue Structure, that list all processes that have access to the 32-bit semaphore. This is formatted using **.D OPENQ**.

**pName** points to the semaphore name, when not anonymous.

**pMuxQ** points to a MUXQ structure, that lists any 32-bit MUX wait semaphore address lists that have included this semaphore. In this example we see one MUX list.

The MUX list may be formatted using **.D SEM32**.

Instead of a **pMuxQ**, the MUX semaphore contains a pointer to the semaphore record (**SR Pointer**) and a count of the number of semaphores in the list (**SR Count**).

There is no special formatting command for the **SR Structure** - it has to be view by displaying storage directly. In this case we see then length, flags and three semaphore handles each followed by the user correlator.

For a description of the 32-bit Semaphore Structures, see the [32-bit Semaphore Structures](#) in the **System Reference**.

## .I - Swap in Storage



Page in a [TSD](#) or a Page of Virtual Storage from the Swapper file.

**Syntax:**

```
.I                                addr ~
                                B
                                D

T                                ~
                                B
                                D      slot
```

**Parameters:**

**T**

When specified it requests the Kernel Debugger page in the [TSD](#) for a specified thread slot.

One TSD is assigned to each thread slot. If the registers for an out-of-[context](#) thread need to be examined then it may be necessary to swap in the TSD for that slot, since the ring 3 stack frame is stored in the TSD when a thread enters the Kernel. The presence or absence of the TSD for a given slot may be deduced by the presence or absence of a value for the **Disp** field of the [.P](#) command.

If the **T** option is omitted it requests the Kernel Debugger page in the page of virtual storage that encompasses the specified **addr**.

**B**

When specified requests that all [breakpoints](#) be re-instated, including those which the current **CS:EIP** may be addressing.

[This parameter is effectively obsolete since breakpoints at the current CS:EIP are correctly handled by the Breakpoint Commands.](#)

**D**

Specifies that a page-in request be scheduled for the Kernel Debugger Daemon thread to execute. In most cases a page-in operation may be performed synchronously, but under the following conditions it is prohibited:

When an interrupt is being handled (that is, not at task time)

When a swapping operation is pending (**TCBfswapping** (**TCB** + 0x1a1) not 0)

When a the current thread is blocked (**TK\_WF\_SLEEPING** (0x40) is set in **TCBWakeFlags** (**TCB** + 0x162))

When in ring0 and InDos is 0

When one of these conditions occurs the page-in request may be scheduled for execution asynchronously by the Debugger Daemon thread by use of the **D** parameter. If the request is successfully scheduled the user is invited to enter the **G** command. The system will dispatch the Daemon thread, in time, which will attempt the page-in request. The Daemon returns control to the debug console using an **INT 3** interrupt.

[addr](#)

This specifies the virtual address of the page to be paged in. The address is effectively rounded down to the nearest 4K page boundary.

**Note:** A selector:offset address specification can only be used if the selector does not reference the packed area of the LDT. If it does then a linear address must be supplied by the user.

[slot](#)

Specifies the thread slot number of the TSD to be paged in. The default slot is the current slot of the debugger's default slot if overridden with the [.S](#) command.

**Results & Notes:**

When an asynchronous page-in is requested the Kernel Debugger will prompt the user with one of the following:

```
task|addr %nnnn|%nnnnnnnn, LDT entry address %nnnnnnnn queued, G to continue
```

task|addr %nnnn|%nnnnnnnnn queued, G to continue

TSD for slot s queued, G to continue

depending upon combination of parameters specified.

On successful completion of a synchronous page-in the user will be prompted with the command prompt.

If .I is unable to complete the request the OS/2 system error code will be displayed (in decimal) in the following message:

OS/2 error code nt

refer to the **Control Programming** reference or to **bseerr.h** C header file for an interpretation of the error code.

## .H - Display Dump File Header Information



Display dump file header information saved by the stand alone dump program in the first sector (512 bytes) of the dump file.

**Syntax:**

.H

**Parameters:**

None.

**Results & Notes:**

This command displays the following information:

```
-----
.h
Dump File Header Info:
  Start Addr1: 0
    End Addr1: 2623213
  Total Disks: 9
    Flag: 11
  Ending addresses by disk:
    2623213      6634079      9846551      12950323
    14965147     17345751     19711393
    22092095     25165823
# -----
```

Each of the fields displayed has the following meaning:

- Start Addr1:** The lowest physical address dumped.
- End Addr1:** The highest physical address dumped.
- Total Disks:** The number of disk volumes the dump data set spans. When the dump is taken to hard disk then the number of volumes is one.
- Flag:** Indicates whether the dump file required decompressing. **0** indicates a compressed dump and **11** a decompressed

dump.

#### Ending Addresses by disk

Shows the range of physical memory dumped to each disk volume.

## .I (DF) - Show Dump State



Display the dump state.

#### Syntax:

.I

#### Parameters:

None.

#### Results & Notes:

This command displays the following summary information:

```
-----
#.i
PROCESS slot:1c Pid:0003 Ord:0013
PTDA    handle=0088 address=%7bcd5844
MTE     handle=018a address=%fdfadf78 (PMSHL32)
SMTE    address=%fc9a4c48
LDT     handle=0187 address=%7a597000
CODE:   user (cs:eip)#005b:17d679f2 cbargs=
STACKS: user (ss:esp)#0053:01382cbc(active)
        ring2(ss:esp)#09be:00004000(bottom)
        ring0 tcbframe=%7bbd4f54 bottom=%7bbd4f9c
-----
```

Each of the fields display has the following meaning:

#### slot:

The current [thread slot](#) at the time the dump was taken. This value is taken from the **TaskNumber** global variable.

#### Pid:

The current [pid](#) when the dump was taken. This value is taken from the **Pid** global variable.

#### Ord:

The [Tid](#) of the current thread at the time the dump was taken. This value is taken from the **TCBOrdinal** (TCB+ 0x0) of the current **TCB**.

#### handle=

The [VMOB](#) handle that represents the control block named to the left.

.I displays this information for the **PTDA**, **MTE**, **SMTE** and **LDT** associated with the current thread when the dump was taken.

See the [.MO command](#) form more information.

#### address=

The address of the object whose name and handle are given on the same line of display.

#### user (cs:eip)

The current user **CS:EIP** when the dump was taken. See the [.R command](#) for related information.

#### cbargs=

The call gate argument count if the current task has made a privilege level transition. See the [.PU command](#) for further information.

**user (ss:esp)**

The current user **SS:ESP** when the dump was taken. See the [.R command](#)

**ring2(ss:esp)**

The current ring 2 **SS:ESP** as saved in **TCBCpl2\_SS** (**TCB** + 0x1bc) and **TCBCpl2\_ESP** (**TCB** + 0x1b8) fields of the current **TCB**.

**ring0 tcbframe=**

The current (or last) kernel entry stack frame pointed to be **TCB\_pFrameBase** (**TCB** + 0x3c) when the current thread made a call or transition to the kernel.

**bottom=**

The base of the ring 0 stack (in its all contexts addressable form) for the current thread.

---

## .K - Display User Stack Trace



Display the user stack-trace for a given [thread slot](#).

**Syntax:**

```
.K          #
.KS         *
.KB         slot
```

**Parameters:**

**.K**

Display stack frame trace assuming the default operation size from the descriptor associated with the code selector of the [user registers](#) for the specified slot.

**.KS**

Display frame trace assuming an operation size of 16-bits (small-model).

**.KB**

Display frame trace assuming an operation size of 32-bits (big-model).

***slot***

Display stack trace for [thread slot](#) *slot*.

The following short-hand may be used for the slot number:

\*                      The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

`_TaskNumber`

#                      The debugger default thread slot. This defaults to the current slot unless overridden by the [.S command](#).

If no slot number is given then all thread slots are displayed, grouped by process.

**Results & Notes:**

The **.K** command operates as a [K command](#) but with the starting stack frame and code segment address implicitly determined from the user's register as displayed by the [.R command](#).

The output from the **.K** command displays exactly as the **K** command but with the slot-number prefixed to the return address when an out-of-context stack trace is displayed. See example output below.

**Warning:**

The **.K** command is subject to the same limitations as noted for the **K** command. See the **K command** description for details.

Example output from an out-of-context stack trace:

```
-----
##.S 8
##.K 37
0037|a6e7:0000006f 03d4 0000 00c5 006f
0037|a6e7:00000000 0000 0000 0000 0000
##-----
```

# .LM - Format Loader Structures (MTE, SMTE, OTE and STE)



Display selected information from the **MTE** and **SMTE** of one or more loaded modules. Optionally format the associated **STE** or **OTE**.

**Syntax:**

```
.LM O I hmt e
      L addr
      P name
      V
      X
```

**Parameters:**

- O** Format information about each object of each load module.  
For 32-bit modules selected fields from the Object Table Entry (OTE) are displayed.  
For 16-bit modules selected fields from the Segment Table Entry (STE) are displayed.
- I** Select Installable File System Driver modules only.
- L** Select Dynamic Link Library modules only. (This includes DLLs and any other modules which are not specifically selectable by the other options.)
- P** Select Physical Device Drivers modules only.
- V** Select Virtual Device Drivers modules only.
- X** Select Executable modules (.EXE) only.
- hmt e** Specifies the handle of the **memory object** assigned to the MTE structure to be formatted.
- addr** Specifies the address of the MTE to be formatted.
- name** Specifies the name (excluding the file extension and path). The MTE matching this name will be formatted. The name



must be specified as a quoted string.

This option requires the SMTE to be present in storage. See below for information on how to make the SMTE present.

The default specification is to scan the entire MTE chain without formatting corresponding STEs or OTEs.

### Results & Notes:

From fix pack 29 for Warp V3 and base Warp V4 the following changes have been made:

.LM has been fixed for the dump formatter so that it displays the short name of modules when the SMTE is swapped out.

the I parameter has been introduced.

with some previous version of the Dump Formatter the .LMP and .LMV commands did not always display output.

From OS/2 Warp V3.0 fix pack 40, OS/2 Warp V4.0 fix pack 10 and OS/2 Warp E-Server the swappable structures referenced by .LM can be forced to be allocated from resident memory by using the **OTE** option of the [RASKDATA CONFIG.SYS statement](#). This will avoid the possibility that .LM might not be able to display information some modules.

The MTE chain is scanned from global symbol:

`_mte_h`

When OTE/STE formatting is not requested output appears as follows:

```
.lm
hmte=0293 pmte=%fdfd1a38 mflags=06903140 e:\os2tools\mrfile32.exe
hmte=027f pmte=%fdfd1c80 mflags=06903142 !pulse
hmte=0272 pmte=%fdfd1db4 mflags=06903152 c:\os2\cmd.exe
hmte=00a0 pmte=%fe0177a8 mflags=0698b194 c:\os2\dll\display.dll
hmte=017a pmte=%fe015abc mflags=0698b198 c:\os2\dll\bvhwndw.dll
hmte=010e pmte=%fef282dc mflags=0691b180 ???
hmte=0101 pmte=%fe016b6c mflags=0691b180 ???
hmte=00f9 pmte=%fe016cd4 mflags=0691b180 c:\os2\mdos\vdma.sys
hmte=00f5 pmte=%fe016de0 mflags=0691b180 c:\os2\mdos\vbios.sys
hmte=0072 pmte=%fff2c919 mflags=0002b180 mvdm.dll
hmte=0006 pmte=%fff2bde0 mflags=0000b980 doscalls.dll
hmte=01c8 pmte=%fdf45e78 mflags=0698b1c8 c:\os2\dll\times.fon
hmte=01c6 pmte=%fe017718 mflags=0698b1c8 c:\os2\dll\helv.fon
hmte=00d5 pmte=%fdf32e60 mflags=0608f1ca c:\os2\pmdm.sys
hmte=00d6 pmte=%fdf32f04 mflags=0608f1c9 c:\os2\dos.sys
hmte=00cd pmte=%fdf49f64 mflags=0608f1c9 c:\os2\testcfg.sys
hmte=00cc pmte=%fdf4fb40 mflags=0628a1c9 c:\os2\hpfs.ifs
hmte=00a2 pmte=%fdf45fb8 mflags=0408e1c9 c:\os2dasd.dmd
hmte=00a1 pmte=%fdf32f8c mflags=0408e1c9 c:\ibm2scsi.add
hmte=009f pmte=%fdf2ff18 mflags=0408e1c9 c:\ibm2flpy.add
hmte=0096 pmte=%fdf41f60 mflags=0408e1c9 c:\print02.sys
hmte=0093 pmte=%fdf2efb8 mflags=0408e1c9 c:\clock02.sys
#
```

The fields formatted have the following meaning:

#### hmte

Handle of the [memory object](#) occupied by this MTE. Taken from *mte\_handle*

#### pmte

Linear address of this MTE

#### mflags

Flag field 1 taken from *mte\_flags1*. These flags have the following interpretation:

name	bit mask	description
NOAUTODS	0x00000000	No Auto DS exists
SOLO	0x00000001	Auto DS is shared
INSTANCEDS	0x00000002	Auto DS is not shared

INSTLIBINIT	0x00000004	Per-instance Libinit
GINISETUP	0x00000008	Global Init has been setup
NOINTERNFIXUPS	0x00000010	internal fixups in .EXE-.DLL applied
NOEXTERNFIXUPS	0x00000020	external fixups in .EXE-.DLL applied
CLASS_PROGRAM	0x00000040	Program class
CLASS_GLOBAL	0x00000080	Global class
CLASS_SPECIFIC	0x000000C0	Specific class, as against global
CLASS_ALL	0x00000000	nonspecific class - all modules
CLASS_MASK	0x00000000	
MTEPROCESSED	0x00000100	MTE being loaded
USED	0x00000200	MTE is referenced
DOSLIB	0x00000400	set if DOSCALL1
DOSMOD	0x00000800	set if DOSCALLS
MTE_MEDIAFIXED	0x00001000	File Media permits discarding
LDRINVALID	0x00002000	module not loadable
PROGRAMMOD	0x00000000	program module
DEVDRVMOD	0x00004000	device driver module
LIBRARYMOD	0x00008000	DLL module
VDDMOD	0x00010000	VDD module
MVDMMOD	0x00020000	Set if VDD Helper MTE (MVDM.DLL)
INGRAPH	0x00040000	In Module Graph
GINIDONE	0x00080000	Global Init has finished
MTEADDRALLOCED	0x00100000	Allocate specific or not
FSDMOD	0x00200000	FSD MTE
FSHMOD	0x00400000	FS helper MTE
MTELONGNAMES	0x00800000	Module supports long-names
MTE_MEDIACONTIG	0x01000000	File Media contiguous memory req
MTE_MEDIA16M	0x02000000	File Media requires mem below 16M
MTEIOPLALLOWED	0x04000000	Module has IOPL privilege
MTEPORTHOLE	0x08000000	porthole module
MTEMODPROT	0x10000000	Module has shared memory protected
MTENEWMOD	0x20000000	Newly added module
MTEDLLTERM	0x40000000	Gets instance termination
MTESYMLOADED	0x80000000	Set if debugger symbols loaded

*name*

The full path name for the module is displayed to the right of the **mflags** field. The name is taken from the `smte_path` of the SMTE. If the SMTE is swapped out then the the name is taken from `mte_modname` (the .DEF file link edit name) and prefixed with an ! symbol.

Where no path information is given then the module is predefined by the system and does not exist separately as a load module file.

The STE and OTE are displayed when the **O** option is specified. These tables are accessed from the address at SMTE+0x1c. This requires that the SMTE be present in storage. If it is not then the following is returned:



???



swappable MTE - swapped

To page in the SMTE use **.LM** without parameters to obtain the MTE address from the **pmte** field. The SMTE address is at MTE + 0x4. Use the **.I command** to page in the SMTE storage.

Under the Dump Formatter nothing can be done, however use of the **OTE** option of the **RASKDATA CONFIG.SYS statement** will guarantee that structures used by **.LM** are retained in resident memory.

For a 16-bit module the STE is formatted as follows:

```
#.lmo 'hpfs'

hmte=00cc pmte=%fdf4fb40 mflags=0628a1c9 c:\os2\hpfs.ifs
seg  sect psiz vsiz hob sel flags
0001 0003 eb24 eb24 0000 0668 8d60 code shr prel rel
0002 0079 d22f d230 0000 0670 8d60 code shr prel rel
0003 00e3 07b5 07b8 0000 0678 8d60 code shr prel rel
0004 00e8 0d8a 0d8c 0000 0680 8d60 code shr prel rel
0005 00f0 0d6e 19c2 0000 0688 8d41 data prel rel
0006 00f7 03fb 03fc 0000 0690 8c41 data prel
0007 00f9 0084 0084 0000 0698 8d41 data prel rel
0008 00fa 0010 0014 0000 06a0 8d41 data prel rel
0009 00fb 0238 0238 0000 06a8 8d41 data prel rel
#
```

The STE fields formatted have the following meaning:

- seg** Segment number. This is a sequential index of module segments. Index entries appearing in the link-edit map will correspond with these values.
- sect** (ste\_offset) Offset in file to segment data.
- psiz** (ste\_size) File data size
- vsiz** (ste\_minsiz) Minimum allocation size
- hob** (ste\_seghdl) [Memory object](#) handle of segment data.
- sel** (ste\_selector) Selector assigned to this segment.
- flags** (ste\_flags) Segment type and attribute flags. These interpretations of these are displayed to the right of the flag word. They are assigned as follows:

<i>name</i>	<i>bit mask</i>	<i>.lmo msg</i>	<i>description</i>
STE_CODE	0x0000	code	code segment type
STE_DATA	0x0001	data	data segment type
STE_PACKED	0x0002		segment is packed
STE_SEMAPHORE	0x0004		segment semaphore
STE_ITERATED	0x0008	iter	segment data is iterated

STE_WAITING	0x0010	move	segment is waiting on semaphore
STE_SHARED	0x0020	shr	segment can be shared
STE_PRELOAD	0x0040	prel	segment is preload
STE_ERONLY	0x0080	EO	execute only if code segment
STE_ERONLY	0x0080	RO	read only if data segment
STE_RELOCINFO	0x0100	rel	if segment has reloc records
STE_CONFORM	0x0200	conf	segment is conforming
STE_RING_2	0x0800	iopl	ring 2 selector
STE_RING_3	0x0C00		ring 3 selector
STE_HUGE	0x1000	disc	huge segment
STE_PAGEABLE	0x2000		just a page can be faulted in
STE_PRESENT	0x2000		packed segment already loaded
STE_SELALLOC	0x4000		used to indicate sel allocated
STE_GDTSEG	0x8000		used to indicate GTD sel alloc

For a 32-bit module the OTE is formatted as follows:

```
#.lmo 'doscall11'

hmte=00a7 pmte=%fdf59f58 mflags=0698b594 c:\os2\dll\doscall11.dll
obj  vsize  vbase  flags  ipagemap  cpagemap  hob  sel
0001 00001354 1a010000 80009025 00000001 00000002 00ad d00e r-x shr alias iopl
0002 0000cde8 1a020000 80002025 00000003 0000000d 00ac d017 r-x shr big
0003 00001844 1a030000 80001025 00000010 00000002 00ab d01f r-x shr alias
0004 000002ce 1a040000 80001025 00000012 00000001 00aa d027 r-x shr alias
0005 000054d0 1a050000 8000d025 00000013 00000006 00a9 d02e r-x shr alias conf iopl
0006 00000270 1a060000 80001023 00000019 00000001 00a8 d037 rw- shr alias
0007 00001b40 1a070000 80001003 0000001a 00000002 0000 d03f rw- alias
```

The OTE fields formatted have the following meaning:

#### obj

Object number. This is a sequential index of module object. Index entries appearing in the link-edit map will correspond with these values.

#### vsize

(ote\_size) Object virtual size

#### vbase

(ote\_base) Object base virtual address

#### flags

(ote\_flags) Attribute flags. The interpretations of these are displayed to the right of the each line. They are assigned as follows:

<i>name</i>	<i>bit mask</i>	<i>.lmo</i>	<i>description</i>
		<i>msg</i>	
OBJREAD	0x00000001	r	Readable Object
OBJWRITE	0x00000002	w	Writeable Object
OBJEXEC	0x00000004	x	Executable Object
OBJRSRC	0x00000008	rsrc	Resource Object
OBJDISCARD	0x00000010	disc	Object is Discardable

OBJSHARED	0x00000020	shr	Object is Shared
OBJPRELOAD	0x00000040	prel	Object has preload pages
OBJINVALID	0x00000080	inv	Object has invalid pages
OBJZEROFIL	0x00000100	zfill	Object has zero-filled pages
OBJRESIDENT	0x00000200		Object is resident
OBJALIAS16	0x00001000	alias	16:16 alias required
OBJBIGDEF	0x00002000	big	Big/Default bit setting
OBJCONFORM	0x00004000	conf	Object is conforming for code
OBJIOPL	0x00008000	iopl	Object I/O privilege level
OBJMADEPRIV	0x40000000		Object is made private for debug
OBJALLOC	0x80000000		Object is allocates used by the loader

#### ipagemap

(ote\_pagemap) Object page map index.

#### cpagemap

(ote\_mapsize) Number of entries in object page map.

#### hob

(ote\_seghdl) [Memory object](#) handle of object data.

#### sel

(ote\_selector) Selector assigned to this object.

If either the segment table or object is not in storage then the following message is issued:

```
%nnnnnnnnx - swapped
```

-----

## .M - Format Memory Structures



Format memory management structures (VMOB, VMAR, VMAL, VMCO, VP and PF).

#### Syntax:

```
.M
    A      options
    O
    C
    L
    V
    P
```

#### Parameters:

##### A

Format Memory Arena Records ([VMARs](#)). See [.MA command](#) for more information.

##### O

Format Memory Object Records ([VMOBs](#)). See [.MO command](#) for more information.

C

Format Memory Context Records (VMCOs). See [.MC command](#) for more information.

L

Format Memory Alias Records (VMALs). See [.ML command](#) for more information.

V

Format Virtual Page structures (VPs). See [.MV command](#) for more information.

P

Format Page Frame structures (PFs). See [.MP command](#) for more information.

*options*

See the corresponding **.Mx** command for details of applicable options.

Prior to fix pack 29 for Warp V3, the **.M** command defaults to:



**.MAMC**



**.MAAMC**

From fix pack 29 of Warp V3 and base Warp V4 the **.M** command is consistent for both Kernel Debugger and Dump Formatter, and defaults to **.MAMC**

For further details see the **M** option of the [.MA command](#).

# **.MA - Format Memory Arena Records (VMAR)**



Display memory arena records (VMARs). Optionally format related object records (VMOBs), alias records (VMALs) and context records (VMCOs).

**Syntax:**

```
.MA M A C maddr ~
      A
      B F
          L R C
          H
      har
      laddr L n
```

**Parameters:**

**A**

This option is used with (and implies) the **M** option. It causes the a match for private area addresses to be made across all [contexts](#). See the **M** option for further details.

**Note:**

Under Kernel Debugger the default is to match addresses in the current context only.

Under Dump Formatter address matches are made across all contexts, that is the **A** option is in permanent

effect.

**B**

Display in-use (busy) arena records in sequential order.

**C**

Display chained memory structures.

Chaining causes related memory structures to be displayed in groups, the head of which is indicated by an \* suffix. The related structures are:

aliases to the associated arena record (**VMALs**).

arena records of all associated alias records (**VMARs**).

shared instance data objects for all related arena records

context records for shared objects of all associated arena records (**VMCOs**). See **.MC command**.

object records of all associated arena records (**VMOBs**). See **.MO command**.

**F**

Display free arena records.

**H**

Follow the arena hash chain pointer. The hash chain is used by virtual memory management to look up a memory object for a given **context** from a linear address. The algorithm proceeds as follows:

The linear address is bitwise ANDed with the hash table mask obtained from **at\_IHashNumbMask (VMAT+0x14)**. The result is shifted right by the allocation granularity for the arena. This is obtained from **at\_IHashNumbShift (VMAT+0x18)**. The result provides an index into the hash table, which is a table of arena handles that head each hash chain. The hash table address is obtained from **ah\_paharHash (VMAH+0x14)** and the **VMAT** address is obtained from **ah\_pat (VMAH+0x18)**.

For OS/2 Warp V3.0, the hashing algorithm amounts to the following:

System Arena:

$\text{index} = (\text{linear address} \gg 0x0c) \& 0x1ff$

Tiled Shared and Private arenas:

$\text{index} = (\text{linear address} \gg 0x10) \& 0x1ff$

VDM Private arenas:

$\text{index} = (\text{linear address} \gg 0x0c) \& 0x1ff$

**L**

Follow the arena forward (left) chain pointer. Arena records for each arena are chained using a double-linked circular chain. The Dump Formatter or Kernel Debugger will not detect wrap-around. This option must therefore be limited by specifying a fixed number of arena records, using the **Ln** operand, or interrupted using **Ctrl+C**.

**M**

Searches for all arena records (of all **contexts**) that represent virtual memory that encloses the address specified in **maddr**. If **maddr** is not specified then the current **CS:EIP** is taken as the matching address. If the storage is in the private arena Kernel Debugger will search the current context only unless the

An **address expression** may be specified. **A** option is specified. The Dump Formatter always searches for matches in all contexts.

**R**

Follow the arena backward (right) chain pointer. Arena records for each arena are chained using a double-linked circular chain. The Dump Formatter or Kernel Debugger will not detect wrap-around. This option must therefore be limited by specifying a fixed number of arena records, using the **Ln** operand, or interrupted using **Ctrl+C**.

**maddr**

Specifies the matching address to be used with the **M** option.

**laddr**

Specifies the linear address of a specific arena record to be formatted.

**Ln**

Specifies the number of arena records to display.

*har*

Specifies the handle of a specific arena record to be formatted.

#### Results & Notes:

Arena records are in contiguous storage, which is anchored from the address given by global variable:

`_parvmOne`

Output from the **.MA** command is formatted using a common template with minor variations.

**Note:** Because a common display template is used for all forms of arena record certain fields will be irrelevant to the records being viewed and may contain garbage information. Specific cases are noted in the examples where this applies.

The following are example of the nine formats of area record:

[Free Arena Record](#)  
[Sentinel Arena Record](#)  
[Boundary Sentinel Arena Record](#)  
[System Arena Records mapped by GDT selectors](#)  
[System Arena Records not mapped by GDT selectors](#)  
[Shared Arena Records for Shared Data](#)  
[Shared Arena Records for Instance Data](#)  
[Private Arena Records for non-shared Data](#)  
[Private Arena Records for shared Data](#)

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

For more examples using of the **.M** family of commands see: [Exploring Memory Management](#).

-----

## Free Arena Record

har	par	cpg	va	flg	next	prev	link	hash	hob	hal	
0263	%fef2948c	000294a2	%00320000	168	0233	0262	0000	0000	02df	0000	=029e
0264	%fef294a2	000294b8	%00000000	000	0000	0000	0000	0000	0000	0000	=0000
0265	%fef294b8	000294ce	%00000000	000	0000	0000	0000	0000	0000	0000	=0000

Free Arena Record Display

#### **Notes:**

Flag bit 0x001 reset signifies a free record.

The only fields of relevance are **har**, **par**, and **cpg**.

Bit positions 0xffe of **flg** and remaining fields may contain garbage from a previous use of the record.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

-----

## Sentinel Arena Record



har	par	cpg	va	flg	next	prev	link	hash	hob	hal
0004	%fef26062	00000000	%60000000	003	0239	0015	0000	0000	ffc0	0000 max=%fffc0000
0005	%fef26078	0000dfb0	%04000000	007	0259	006e	0000	0000	fff0	0000 max=%1fff0000

## Sentinel Arena Records

### Notes:

Flag bit 0x002 set signifies a sentinel record.

**hob** is not relevant to sentinel records. (The value displayed originates from the **max=** field).

For OS/2 2.1, arena record 4 is sentinel for the system arena.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

# Boundary Sentinel Arena Record

har	par	cpg	va	flg	next	prev	link	hash	hob	hal
0005	%fef26078	0000dfb0	%04000000	007	0259	006e	0000	0000	fff0	0000 max=%1fff0000

## Boundary Sentinel Arena Record

### Notes:

Flag bits 0x006 set signify a boundary sentinel record.

The boundary sentinel indicates the boundary between the shared and private arena address spaces. Consequently there is only one boundary sentinel to be found in a system.

**hob** is not relevant to sentinel records. (The value displayed originates from the **max=** field).

For OS/2 2.1, arena record 5 is boundary sentinel for the shared arena.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

# System Arena Record Mapped by GDT

har	par	cpg	va	flg	next	prev	link	hash	hob	hal	
0006	%fef2608e	00000003	%fff20000	009	000f	00b0	0000	0000	0007	0000	sel=0100
0007	%fef260a4	0000000b	%ffe27000	009	0008	001a	0000	0000	0008	0000	sel=0400
0008	%fef260ba	0000000b	%ffe32000	009	0009	0007	0000	0000	0009	0000	sel=0f00
0009	%fef260d0	00000010	%ffe3d000	009	000b	0008	0000	0000	000a	0000	sel=0120

System arena records - address space mapped by a GDT selector

#### Notes:

Flag bit 0x008 set signifies a selector mapping.

**va** value >= that specified in the [System Arena Sentinel](#) signifies system area area record.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

---

## System Arena Record Not Mapped by GDT

har	par	cpg	va	flg	next	prev	link	hash	hob	hal	
000e	%fef2613e	00000001	%fff16000	001	01d9	0083	0000	0000	000f	0000	=0000
000f	%fef26154	00000001	%fff23000	001	0010	0006	0000	0000	0010	0000	=0000
0010	%fef2616a	00000002	%fff24000	001	0011	000f	0000	0000	0011	0000	=0000

System arena records - address space not mapped by a GDTselector

#### Notes:

Flag bit 0x008 set to 0 signifies a no selector mapping.

**va** value >= that specified in the [System Arena Sentinel](#) signifies system area area record.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

---

## Shared Arena Record for Shared Data

har	par	cpg	va	flg	next	prev	link	hash	hob	hal	
00b2	%fef26f56	00000010	%1a0a0000	379	00b6	00b3	0000	0000	00be	0000	hco=001ed
00b3	%fef26f6c	00000010	%1a090000	3d9	00b2	00a9	0000	0000	00c0	0000	hco=001ee
00b4	%fef26f82	00000010	%1a0e0000	379	00bb	00b5	0000	0000	00c1	0000	hco=0022e

Shared arena, shared data.

#### Notes:

Flag bit 0x200 set to 1 signifies shared arena, shared data.

Context records chained from **hco** value will list the processes that currently share the memory object represented by this arena record.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

---

## Shared Arena Record for Instance Data

har	par	cpg	va	flg	next	prev	link	hash	hob	hal	
00e9	%fef27410	00000020	%1abb0000	179	0105	00ea	0000	0000	02b5	0000	=0000
00ea	%fef27426	00000010	%1aba0000	179	00e9	00eb	0000	0000	02b6	0000	=0000

Shared arena, instance data.

#### Notes:

Flag bit 0x200 set to 0 with a **va** value not in the system arena and 0 hptda indicates shared arena instance data.

Object records chained from **hob** value will list the objects and processes that map to the common virtual address range represented by this arena record.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

## Private Arena Record Non-Shared Data

har	par	cpg	va	flg	next	prev	link	hash	hob	hal	
00e7	%fef273e4	00000010	%00020000	179	00e8	00db	0000	0000	011e	0000	hptda=0097
00e8	%fef273fa	00000010	%00030000	179	012f	00e7	0000	0000	011f	0000	hptda=0097

Private non-shared data, process owned arena records

#### Notes:

Arena records not satisfying the criteria for any of the System, Sentinel or Shared Arena records are assumed to be private arena records.

If the private memory object is shared (for example, .EXE code segments running in more than one process) then the associated private arena records for the sharing processes are chained from the **link** field as long as **hal** is zero.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

## Private Arena Record Shared Data

har	par	cpg	va	flg	next	prev	link	hash	hob	hal	
01e2	%fef28976	00000010	%00010000	1c9	01e3	01e0	00db	0000	011c	0000	hptda=0237

Private shared data, process owned arena records

#### Notes:

Arena records not satisfying the above criteria are assumed to be private arena records.

If the private memory object is shared (for example, .EXE code segments running in more than one process) then the associated private arena records for the sharing processes are chained from the **link** field as long as **hal** is zero.

For a description of the fields formatted by **.MA** select [.MA Output Field Descriptions](#)

# .MA Output Field Descriptions

Output from **.MA** is in a tabular format of the following form:

```
har      par      cpg      va      flg next prev link hash hob   hal
0005 %fef26078 0000dfb0 %04000000 007 0259 006e 0000 0000 fff0 0000 max=%1fff0000
0009 %fef260d0 00000010 %ffe3d000 009 000b 0008 0000 0000 000a 0000 sel=0120
00b4 %fef26f82 00000010 %1a0e0000 379 00bb 00b5 0000 0000 00c1 0000 hco=0022e
00ea %fef27426 00000010 %1aba0000 179 00e9 00eb 0000 0000 02b6 0000 =0000
00e7 %fef273e4 00000010 %00020000 179 00e8 00db 0000 0000 011e 0000 hptda=0097
```

The field headings have the following meaning:

- har** The handle of the arena record being formatted. This is the unique identifier by which the arena record is known.
- par** The linear address of the arena record being displayed.
- cpg** The size of the contiguous linear address range reserved by this arena record expressed as a hexadecimal number of pages. This value will be greater then or equal to the size of the memory object that occupies this linear address range.  
  
For small allocations (<64K) in non-system arenas this will usually be rounded to 0x10 pages so that the [CRMA](#) may be applied.  
  
For free records this field is used as a chain pointer to the next free record but only the low order 20 bits are displayed by **.MA**.

**va** The linear address of the beginning of the memory object represented by the arena record.

**flg** Arena record flags. These may take a combination of the following settings:

Name	Bit Mask	Description
AR_INUSE	0x001	Record not on free list
AR_TAG	0x006	Record type mask
AR_TAGREG	0x000	Regular record
AR_TAGSEN	0x002	Sentinel
AR_TAGBSEN	0x006	Boundary sentinel
AR_SELMAP	0x008	Memory mapped by selector
AR_SELBASEALL	0x00c	Base selector map all
AR_SELMASK	0x00c	Selector map mask
AR_RELOAD	0x010	Pre-reserved for huge item or or reserved for reload of MTE object

AR_WRITE	0x020	write permission
AR_USER	0x040	user permission
AR_EXEC	0x080	executable permission
AR_READ	0x100	Read permission
AR_HCO	0x200	Record linked to Context List
AR_GUARD	0x400	guard page
AR_SGS	0x800	Registered under Screen Group Switch control.

#### next

Handle of next arena record within the same arena (private, shared or system).

#### prev

Handle of the previous record within the same arena (private, shared or system).

#### link

Handle of an associated arena record.

For private arena sentinel records this points to the boundary sentinel.

For shared arena shared data this points to other private arenas sharing the same object.

For alias objects this points to the arena record of the associated (aliased) object.

#### hash

Handle of the next arena record whose *va* hashes to the same hash chain.

#### hob

The handle of the associated memory object record. See [.MO](#) command.

#### hal

The handle of the associated memory alias record. See [.ML](#) command. If this field is set to a value of 0xffff then this is not the handle of an alias record, but signifies that this arena has been privatized by the creation of an alias to it.

#### hptda= *hhhh*

The handle of the pseudo-object that is the PTDA of the process that has this arena record assigned to its private address space. Use [.MO](#) *hhhh* to display the PTDA pseudo-object and hence obtain the address of its virtual address.

#### max= *%mmmmmmmm*

Maximum linear address of the area headed by this sentinel record.

#### sel= *ssss*

GTD selector that is assigned to a system arena memory object.

#### hco= *cccc*

Handle of the first context record that represents processes sharing this shared arena, shared data memory object. See [.MC](#) command.

-----

## .MC - Format Memory Context Records (VMCO)



Display memory context records ([VMCOs](#)).

#### Syntax:

```

.MC
      B      F      hco
      C      C      laddr      Ln

```

### Parameters:

**B** Display in-use (busy) alias records in sequential order.

**C** Display chained context records.

Chaining causes related context records that are chained from **hconext** of the current context to be formatted. The head of each group indicated by an \* suffix. Context records are chained to represent each instance of an object being shared among several processes. The head of the chain is pointed to by the **hco** field of the corresponding arena record See [.MA command](#) for information on formatting arena records.

#### **Notes:**

There is no pointer to the arena record from the context record. Associated arena records have to be found by scanning arena or object records.

The **C** option will not format context records from the head of the chain. Do achieve this, locate the corresponding arena record and use

```
.MAC har
```

**F** Display free alias records.

*laddr* Specifies the linear address of a specific context record to be formatted.

*Ln* Specifies the number of context records to display.

*hco* Specifies the handle of a specific context record to be formatted.

### Results & Notes:

Context records are in contiguous storage, which is anchored from the address given by global variable:

```
_pcovmOne
```

Output from the **MC.** command appears in one of two formats:

[Free Context Records](#)  
[Busy Context Records](#)

For a description of the fields formatted by **.MC** select [.MC Output Field Descriptions](#)

For more examples using of the **.M** family of commands see: [Exploring Memory Management](#).

In some versions prior to fix pack 29 for Warp V3 the Dump Formatter did not interpret and format the VMCO correctly. In addition VCMO chains were not followed correctly when the **C** options was used. These problems are fixed from fix pack 29 of Warp V3 and base Warp V4.

## Free Alias Records

```
hco=00313 pco=ffe4bf7a pconext=ffe4bf7f
```

```
hco=00314 pco=ffe4bf7f pconext=ffe4bf84
hco=00315 pco=ffe4bf84 pconext=ffe4bf89
hco=00316 pco=ffe4bf89 pconext=ffe4bf8e
```

#### Free Context Record Display

##### Notes:

**pconext** is used as a chain pointer to the next free context record.

For a description of the fields formatted by **.MC** select [.MC Output Field Descriptions](#)

---

## Busy Context Records

```
hco=00001 pco=ffe4b020 hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe
hco=00002 pco=ffe4b025 hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe
hco=00003 pco=ffe4b02a hconext=00000 hptda=00ae f=13 pid=0002 c:pmshell.exe
hco=00004 pco=ffe4b02f hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe
```

#### Selector Busy Context Records

##### Notes:

Flag bit 0x20 set signifies a context handle > 64k. In effect this is a 1 bit extension of the 16 bit **hco** field of the VMCO. The **.ML** command takes this into account when formatting VMCO records.

Flag bit 0x80 set signifies that the context has been privatized. This implies that the object was originally shared but a private instance of it has subsequently been created. Typically this occurs when DosDebug is used to access a debuggee's shared data.

For a description of the fields formatted by **.MC** select [.MC Output Field Descriptions](#)

---

## .MC Output Field Descriptions

Output from **.MC** appears in one of is of the following forms:

```
hco=00317 pco=ffe4bf8e pconext=ffe4bf93
hco=00001 pco=ffe4b020 hconext=00000 hptda=00ae f=1d pid=0002 c:pmshell.exe
```

Each of the fields has the following meaning:

##### **hco=**

The handle of the context record being formatted. This is the unique identifier by which the context record is known.

##### **pco=**

The linear address of the context record being formatted.

##### **hconext=**

For busy records this is the handle of the next context record that represents another user of the related (shared) object. For free records this is the handle of the next free record.

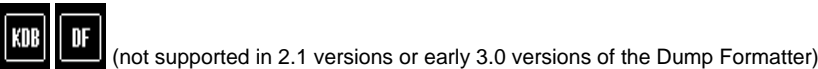
**hptda=** The handle of the PTDA [pseudo-object](#) that represents the process sharing the corresponding memory object.

**f** Context record flags.  
The following flags are defined:

<i>name</i>	<i>bit mask</i>	<i>description</i>
CO_CREATOR	0x01	originating context
CO_PRIV	0x80	Privatized context
CO_HCOH	0x20	Next context record handle > 64k
CO_WRITE	0x02	Write access
CO_USER	0x04	User attribute
CO_EXEC	0x08	Execute access
CO_READ	0x10	Read access
CO_GUARD	0x40	Guard page

**pid=** This names the process Id and process executable that corresponds to the **hptda** field.  
\* EBOOKIE (IPFTAGS.DEF)

## .MK - Display Memory Lock Information Records (VMLKI)



Display memory lock information records.

**Note:**

This command was implemented by [feature 82818](#) for the **ALLSTRICT** and **HSTRICT** kernels only. It is not available is either of the GA versions for OS/2 Warp V3.0 or OS/2 V2.11.

After OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 **.MK** may be enabled under the **RETAIL** kernel by means of the **LOCKS** option of the [RASKDATA CONFIG.SYS statement](#)..

Under the **ALLSTRICT** and **HSTRICT** kernels **.MK** may be disabled by specifying the **NLOCKS** option of the **RASKDATA** statement.

**Syntax:**

```
.MK                                ^
                                hob
                                Ln
```

**Parameters:**

*none* Lists all lock information records for all memory objects with locked records.

**L/n** Specifies the number of lock information records to display for a a given **hob**.



***hob***

Specifies the handle of a specific object record whose lock information records are to be formatted.

#### **Results & Notes:**

Lock information records are maintained for outstanding memory locks in memory lock information records (**VMLIs**) which are located at the address given by global variable:

**\_pVMLIHead**

When a memory lock request is successfully executed a lock handle is returned to the caller for later use when unlocking memory. The lock handle normally resides in the caller's storage. It comprises a concatenation of:

- the requestor's **hptda**
- the **hob** whose pages are being locked
- the page number
- the number of pages
- request flags

In addition a check-sum or signature is calculated from these values and stored with the lock handle.

The **VMLI** is a copy of the constituents of the lock handle that resides in system memory. In addition it includes:

- the requestor's return address
- a pointer to the next **VMLI**
- a pointer to the requestor's lock handle

The **.MK** command formats the contents of the **VMLI** then re-calculates the signature. The calculated and saved signatures should be identical.

Next the lock handle is accessed. If it differs from the corresponding **VMLI** then it too is formatted and the signature is re-calculated and displayed. If either the formatted lock handle and corresponding **VMLI** or the calculated and extracted signatures disagree then a problem may be indicated. For example, an overlaid or freed lock handle. However, there is no requirement for lock requestors to retain their lock handles in their original locations.

#### **Warning:**

Prior to fix pack 29 for Warp V3, the Kernel Debugger can trap when attempting to format lock handles from freed memory. This is fixed in the Kernel Debugger and incorporated into the Dump Formatter in defect 155843 from fix pack 29 for Warp V3 and base Warp V4.

#### **Note:**

When feature 82818 is present **VMLI** records are automatically formatted when displaying memory object records with locked pages, using the **.MO** command.

Output from the **.MK** command appears as follows:

```
-----
##.mk
  pvml  cs    eip    phlock  cpg    va    flg  hptda  hob  sig  csig
%fe679f1c 0170 fffa015d %fd17d480 0001 %013f1000 0003 0091 0424 18aa 18aa
%fe68a1dc 0170 fffa015d %fd17d468 0001 %013f1000 0003 0091 0424 18aa 18aa
%fe74539c 0170 fff3e551 %ffe006ff 0002 %fff33000 0001 02f7 0016 0252 0252
%fe712c54 0170 fff3e551 %ffe00577 0003 %fff38000 0001 02f7 0016 0258 0258
%fe761e0c 0908 00000878 %7b6b7d0c 0001 %ffee9000 0005 0091 0190 011f 011f
          0000 %4c800000 0ff0 0001 0000 0000 d7f5
%fe777e18 0908 00000841 %7b6b7d0c 0006 %ffeea000 0005 0091 0227 01bc 01bc
          0000 %4c800000 0ff0 0001 0000 0000 d7f5
%fe777e3c 0908 00000809 %7b6b7d0c 0001 %ffef0000 0005 0091 022c 01c2 01c2
          0000 %4c800000 0ff0 0001 0000 0000 d7f5
%fe777e60 0908 0000072b %7c224066 0002 %17c40000 0005 0091 0199 7e72 7e72
%fe777e84 0908 0000072b %7c224058 0001 %7a022000 0001 0091 0168 a224 a224
%fe777ef8 0908 000006ee %7c22403c 0001 %7a022000 0001 0091 0168 a224 a224
%fe777f28 0908 000006ee %7c22404a 0002 %17c80000 0005 0091 0196 7eaf 7eaf
```

%fe777f4c 0908 000000a5 %7c22402e 0001 %fe763000 0005 0091 0003 e80c e80c

-----

The field headings have the following meaning:

**pvmli**

Address of the **VMLI** record.

**cs**

Code selector of the requestor of the memory locking function. For calls made through a **DevHlp** request this is taken from **TCBpDHRetAddr** (**TCB** + 0x74). For internal requests the immediate caller of **\_VmLockMem** is displayed.

A blank value indicates information from the lock handle is being formatted, because it does not agree with the corresponding **VMLI**. See note above.

If the lock request was made by a 16-bit caller then the 16-bit far return address is contained within the **eip** only and the **cs** value is the next word from the stack following the return address.

**eip**

The instruction pointer of the requestor of the memory locking function. For calls made through a **DevHlp** request this is taken from **TCBpDHRetAddr** (**TCB** + 0x74). For internal requests the immediate caller of **\_VmLockMem** is displayed.

A blank value indicates information from the lock handle is being formatted, because it does not agree with the corresponding **VMLI**. See note above.

**phlock**

The address of the lock handle buffer supplied with the lock request.

**cpg**

The number of contiguous pages locked.

**va**

The linear address of the first page locked.

**flg**

The flags saved from the lock request.

The following bit settings are defined:

<i>Bit value</i>	<i>Description</i>
0x01	Lock is a long-term
0x02	Verify lock call
0x04	Lock originated from a DevHlp

**hptda**

The **hptda** of the lock requestor.

**hob**

The handle of the associated memory object record, some of whose pages are. See the **.MO** command.

**sig**

The signature value extracted from the **VMLI** or lock handle.

**csig**

The recalculated signature based on information saved in the **VMLI** or lock handle.

For related information see also the [Virtual Memory Lock Trace](#).

-----

## .ML - Format Memory Alias Records (VMAL)



(not supported in early 2.1 versions of the Dump Formatter)

Display memory alias records ([VMALs](#)). Optionally format related arena records ([VMARs](#)), object records ([VMOBs](#)) and context records ([VMCOs](#)).

#### Syntax:

```
.ML [B F C] [hal laddr] Ln
```

#### Parameters:

**B**

Display in-use (busy) alias records in sequential order.

**C**

Display chained memory structures.

Chaining causes related memory structures to be displayed in groups, the head of which is indicated by an \* suffix. The related structures are:

the associated arena record ([VMAR](#)). See [.MA command](#)

aliases to the associated arena record ([VMALs](#)).

arena records of all associated alias records

shared instance data objects for all related arena records

context records for shared objects of all associated arena records ([VMCOs](#)). See [.MC command](#).

object records of all associated arena records ([VMOBs](#)). See [.MO command](#).

**F**

Display free alias records.

*maddr*

Specifies the matching address to be used with the **M** option.

*laddr*

Specifies the linear address of a specific alias record to be formatted.

*Ln*

Specifies the number of arena records to display.

*hal*

Specifies the handle of a specific alias record to be formatted.

#### Results & Notes:

Alias records are in contiguous storage, which is anchored from the address given by global variable:

`_palVMAliases`

From fix pack 29 for Warp V3 and base Warp V4 the Dump Formatter `.ML` command has been enhanced to work independently of kernel symbols. Prior to these releases, kernel symbols were required to enable the Dump Formatter to locate `_palVMAliases`.

Output from the `.ML.` command appears in one of three formats.

[Free Alias Record](#)  
[Selector Alias Record](#)  
[Linear Address Alias Record](#)

For a description of the fields formatted by `.ML` select [.ML Output Field Descriptions](#)

For more examples using of the **.M** family of commands see: [Exploring Memory Management](#).

---

## Free Alias Records

```
hal=000e pal=%ffe61088 palnext=ffe610a0 pgoff=00000 f=000
hal=000f pal=%ffe61090 palnext=ffe61088 pgoff=00000 f=000
hal=0010 pal=%ffe61098 palnext=ffe61090 pgoff=d4460 f=000
```

Free Alias Record Display

### Notes

Flag bit 0x001 reset signifies a free record.

The only fields of relevance are **hal**, **pal**, and **palnext**.

For a description of the fields formatted by **.ML** select [.ML Output Field Descriptions](#)

---

## Selector Alias Records

```
hal=000a pal=%ffe61068 har=0208 cs=0056 ds=d446 cref=001 f=13
hal=000b pal=%ffe61070 har=020b cs=0056 ds=d446 cref=001 f=13
```

Selector Alias Record Display

### Notes

Flag bit 0x02 set signifies a CS alias record.

Flag bit 0x10 set signifies that DS is valid, i.e CS is an alias for the same storage mapped by DS. For example after use of DosCreateCSAlias. This flags distinguishes this form of the alias record

This alias applies to selectors within a specific (process) context.

For a description of the fields formatted by **.ML** select [.ML Output Field Descriptions](#)

---

## Linear Address Alias Records

```
hal=0001 pal=%ffe61020 har=00b8 hptda=009f pgoff=00000 f=001
hal=0002 pal=%ffe61028 har=00b9 hptda=009f pgoff=00000 f=001
```

```
hal=0003 pal=%ffe61030 har=001b hptda=009f pgoff=00000 f=001
```

## Linear Address Alias Record Display

### Notes

Flag bit 0x10 reset distinguishes this form of the alias record.

hptda and har refer to the context and arena which has been aliases.

The context and arena of the creator of the alias may be shown using

```
.MLC hal.
```

For a description of the fields formatted by **.ML** select [.ML Output Field Descriptions](#)

---

## .ML Output Field Descriptions

Output from **.ML** appears in one of is of the following forms:

```
hal=0006 pal=%ffe61048 har=01b8 hptda=009f pgoff=00000 f=021
hal=000a pal=%ffe61068 har=0208 cs=0056 ds=d446 cref=001 f=13
hal=0011 pal=%ffe610a0 palnext=ffe610a8 pgoff=00000 f=000
```

Each of the fields has the following meaning:

<b>hal</b>	The handle of the alias record being formatted. This is the unique identifier by which the alias record is known.
<b>har</b>	The handle of the arena record which represents the aliasing linear address range. The arena record for the aliased linear address is pointed to by the <b>link</b> field of the aliasing <b>har</b> .
<b>hptda</b>	The handle of the PTDA object of the context which has been aliased. This may also take the value of a system owner when memory is in the process of being freed.
<b>pgoff</b>	The page offset into the arena which is aliased. Note: aliases may map partial objects. The number of pages aliased may be determined from the arena record which represents the aliased memory. Use <b>.MLC /a/</b> to display this information.
<b>cs</b>	The Code Selector created to alias R/W memory.
<b>ds</b>	The Data Selector which has been aliased by a Code selector.
<b>cref</b>	The reference count for the number of time a Code Selector alias has been requested for a given Data Selector.
<b>palnext</b>	The pointer to the next free alias record.
<b>f</b>	Alias record flags. For Selector Aliases this is a 6-bit field, for other aliases this is a 12-bit field.

The following flags are defined:

<i>name</i>	<i>bit mask</i>	<i>description</i>
AL_ISBUSY	0x1	Set if record is busy
AL_CSALIAS	0x2	Set is CS alias record
AL_MEMMAP	0x4	Set if MemMapAlias Record
AL_DBGALIAS	0x8	Set if Debug alias
AL_CSDSVALID	0x10	Set if DS selector valid
AL_DEVHLP	0x20	Set if Devhlp Alias
AL_PRIV	0x40	Set if privatized alias
AL_VDM	0x80	Set if VMD alias
AL_NOALIAS	0x100	Set if UVIRT mapping in VDMs

# .MO - Format Memory Object Records (VMOB)



Display memory object records ([VMOBs](#)). Optionally format related alias records ([VMALs](#)), arena records ([VMARs](#)) and context records ([VMCOs](#)).

If [feature 82818](#) is installed, then under the Kernel Debugger only, lock information records will be formatted whenever a memory object with locked pages is displayed. See the [.MK command](#) for more information.

**Syntax:**

```
.MO      V      M      maddr      ,
                                     ,
                                     B      C      hob
                                     F      laddr      L n
                                     N
                                     P
                                     S
```

**Parameters:**

- B** Display in-use (busy) object records in sequential order.
- C** Display chained memory structures.  
  
Chaining causes related memory structures to be displayed in groups, the head of which is indicated by an \* suffix. The related structures are:
  - the associated arena record ([VMAR](#)). See [.MA command](#)
  - aliases to the associated arena record ([VMALs](#)).
  - arena records of all associated alias records
  - shared instance data objects for all related arena records

context records for shared objects of all associated arena records (VMCOs). See [.MC command](#).

object records of all associated arena records (VMOBs).

Note: Pseudo-objects have no related memory objects.

**F**

Display free object records.

**M**

Searches for a [pseudo-object](#) whose address matches the address specified for *maddr*. If no match is found then nothing is displayed.

**Notes:**

The pseudo-object address specified must be an exact match for hit.

The pseudo-object address is that of the object itself and not the VMOB that represents it.

A *selector:offset* form of address may not be specified. However a physical address may be specified in order to bypass virtual address validation done by Kernel Debugger and Dump Formatter.

**N**

Specifies that only normal object records be displayed. These are objects whose linear address allocation is represented by an arena record. Contrast this with [Pseudo-Object](#) and [System Object](#). See also [.MA command](#) for details of arena record display.

**P**

Specifies that [pseudo-object](#) records be displayed.

**S**

Specifies that objects to be displayed are those whose memory management semaphore is busy or wanted. The memory management semaphore is used internally for serialising access to memory management structures. It should not be confused with the memory locking as provided by the DevHlp\_Lock, DevHlp\_Unlock, DevHlp\_VMLock and DevHlp\_VMUnlock calls.

**V**

Specifies verbose mode of display. The address of the VMOB structure is displayed but object description and owner interpretation is suppressed.

*maddr*

Specifies the matching address to be used with the **M** option.

An [address expression](#) may be specified.

*laddr*

Specifies the linear address of a specific object record to be formatted.

An [address expression](#) may be specified.

**L/n**

Specifies the number of object records to display.

[/hob](#)

Specifies the handle of a specific object record to be formatted according to the criteria specified by the other options.

**Results & Notes:**

Object records are located in contiguous storage, which is anchored from the address given by global variable:

`_pobvmOne`

Output from the **.MO.** command appears in one of four formats:

[Normal Object](#)  
[Pseudo-Object](#)  
[Free Object Record](#)  
[System Object](#)

From fix pack 29 for Warp V3 and base Warp V4 the output of this command has been enhanced by adding **hnte** interpretation for system

objects. The appears on the right hand end of the command output following the textual interpretation of the **own** field.

From fix pack 35 for Warp V3 and base Warp V4 Device Driver and File System Driver allocated objects will record the driver's true **hmte** as its **hmte** owner Id instead of the one of the system objects **dd1-dd16** and **fsd1-fsd8**.

Prior to fix pack 29 for Warp V3 the **.MO** command could trap D while formatting MTE objects. This is fixed from fix pack 29 for Warp V3 and base Warp V4.

For a description of the fields formatted by **.MO** select [.MO Output Field Descriptions](#)

For more examples using of the **.M** family of commands see: [Exploring Memory Management](#).

-----

## Normal Object Records

hob	har	hobnxt	flgs	own	hmte	sown,cnt	lt	st	xf	
000d	000c	0000	0325	ffba	0000	0000 00	00	00	00	lock
000e	000d	0000	0000	ffaa	0006	0000 00	00	00	00	os2krnl
000f	000e	0000	0000	ffaa	0006	0000 00	00	00	00	os2krnl
0010	008f	0000	402c	00ae	0115	0000 00	00	00	00	priv 0002 c:\pmsshell.exe
0011	0010	0000	0000	ff37	0000	0000 00	00	00	00	romdata

### Normal Object Record Display

hob	pob	har	hobnxt	flgs	own	hmte	sown,cnt	lt	st	xf
0001	%fec80020	0001	fec8	0000	fff1	0000	0000 00	00	00	00
0002	%fec80030	0002	fec8	0000	ffe3	0000	0000 00	00	00	00
0003	%fec80040	0003	fec8	0000	ffec	0000	0000 00	01	00	00

### Normal Object Record Display - verbose form

#### Notes

The verbose form is specified using the **Voption**. This causes the suppression of **owner** and **hmte** interpretation.

For a description of the fields formatted by **.MO** select [.MO Output Field Descriptions](#)

-----

## Pseudo-Object Records

hob	va	flgs	own	hmte	sown,cnt	lt	st	xf	
0004	%fff13238	8000	ffe1	0000	0000 00	00	00	00	vmah
0005	%fff13190	8000	ffe1	0000	0000 00	00	00	00	vmah
0006	%fff0a891	8000	ffa6	0000	0000 00	00	00	00	mte doscalls.dll
0072	%ffe3c7d4	8000	ffcb	0000	0000 00	00	00	00	ptda 0001 *sysinit
007a	%fff0b3fa	8000	ffa6	0000	0000 00	00	00	00	mte mvdn.dll
007b	%fff0b26b	8000	ffa6	0000	0000 00	00	00	00	mte fshelper.dll
0091	%fe7349ac	8000	ffa6	0000	0000 00	00	00	00	mte c:\pmshapim.dll
0098	%7b9e4060	8000	ffe1	0000	0000 00	00	00	00	vmah
009d	%fe722fb8	8000	ffa6	0000	0000 00	00	00	00	mte c:\clock02.sys



Pseudo-Object Record Display

Notes

The 0x8000 flag bit signifies as [pseudo-object](#).

For a description of the fields formatted by **.MO** select [.MO Output Field Descriptions](#)

Free Object Records

hob	va	flgs	own	hmte	sown,cnt	lt	st	xf
02b5	%fec82b70	0000	0000	0000	0000 00	00	00	00 free
02b6	%fec82b80	0000	0000	0000	0000 00	00	00	00 free
02b7	%fec82b90	0000	0000	0000	0000 00	00	00	00 free
02b8	%fec82ba0	0000	0000	0000	0000 00	00	00	00 free
02b9	%fec82bb0	0000	0000	0000	0000 00	00	00	00 free

Free Object Record Display

Notes

- Flag bit 0x001 reset signifies a free record.
- The only fields of relevance are **va** and **pob** when the **V** option is specified.
- The **va** field is used a link field to other free VMOBs.

For a description of the fields formatted by **.MO** select [.MO Output Field Descriptions](#)

System Object IDs

fff0 vmllock  
fff1 vmob  
fff2 vmsgs  
fff3 vmbmp16

Example System Object Display

Notes

- [System object IDs](#) are not represented by VMOB structures. They are pre-defined IDs for system components.
- The Dump Formatter and Kernel Debugger display only object names when displaying system objects.

# .MO Output Field Descriptions

Output from **.MO** appears tabular for with one of the following headings:

```
hob      va      flgs own  hmte  sown,cnt  lt  st  xf
0004  %fff13238  8000 ffe1 0000  0000 00  00 00 00 vmah
0006  %fff0a891  8000 ffa6 0000  0000 00  00 00 00 mte      doscalls.dll

hob      pob      har hobnxt flgs own  hmte  sown,cnt  lt  st  xf
0003  %fec80040  0003 fec8 0000 ffec 0000  0000 00  01 00 00
0004  %fec80050  %fff13238 8000 ffe1 0000  0000 00  00 00 00

hob      har hobnxt flgs own  hmte  sown,cnt  lt  st  xf
0001  0001 fec8 0000 fff1 0000  0000 00  00 00 00 vmob
0002  0002 fec8 0000 ffe3 0000  0000 00  00 00 00 vmar
```

Each of the heading fields has the following meaning:

- hob**                      The handle of the object record being formatted. This is the unique identifier by which the object record is known.
- hobnext**                The handle of next shared instance data object that maps to the same linear address range (shares the same arena record but maps to a different physical address).
- har**                     The handle of the arena record that describes the linear address range allocated to this object.
- pob**                     The linear address of the object record.
- va**                      The virtual address of the [pseudo-object](#) named in the object description.
- flgs**                    Object record flags.
- The following flags are defined:

<i>Name</i>	<i>Bit Mask</i>	<i>Description</i>
OB_PSEUDO	0x8000	Pseudo-object
OB_API	0x4000	API located object
OB_LOCKWAIT	0x2000	Waiting for a lock conflict to resolve
OB_LALIAS	0x1000	Object has aliases
OB_SHARED	0x0800	Object's contents are shared
OB_UVIRT	0x0400	UVirt object
OB_ZEROINIT	0x0200	Object is zero-initialised
OB_RESIDENT	0x0100	Initial allocation was resident
OB_LOWMEM	0x0040	Object is in low memory
OB_GUARD	0x0080	Guard page attribute
OB_EXEC	0x0020	Executable attribute
OB_READ	0x0010	Readable attribute
OB_USER	0x0008	User attribute

OB_WRITE	0x0004	Writeable attribute
OB_HUGE	0x0002	Object is huge
OB_SHRINKABLE	0x0001	Object is Shrinkable (only if also OB_SHARED)
OB_DHSETMEM	0x0001	DevHlp_VMSetMems are allowed the object

#### Notes:

See [Pseudo-Objects](#) when OB\_PSEUDO is set.

OB\_API is set as a result of allocation made by some APIs (for example, DosExecPgm). It forces page alignment and signals a likelihood of long-term locking.

OB\_HUGE is set when the object is created by DosAllocHuge API.

When OB\_LOCKWAIT is set then the thread has detected a lock request conflict and wishes to wait for the conflict to resolve. The conflict occurs because a contiguous storage lock has been requested but cannot be satisfied because one or more of the pages are already short-term locked. If the current request is for a short-term lock then the thread will wait up to 10 seconds for the conflict to clear. If the time-out expires then the current short-term lock request ends in error and the following message appears on the debugger screen:

```
VMLOCK: Short term lock for > 10 secs: hob=hob
```

If the current request is for a long-term lock then the thread will wait indefinitely. In both cases the [block Id](#) the thread waits on is the address of the VMOB flag word (VMOB+0x4). See [.PB command](#) for information on thread slots waiting on Block Ids.

#### own

This is the **hob** of the owner of this object. The owning **hob** may be in one of three categories:

##### 1. [System Owner](#)

Used to indicate system owned objects. The owner description usually indicates the type of object that is being displayed. For example, the LDT for process 9 running pulse.exe is owned by system object **0xffb9** and has a description

**ldt 0009 c:pulse.exe.**

##### 2. Module Owner ([hmte](#))

This is used for shared (code and data) objects that are part of a load module file. The **.hmte** of the associated load module is used as the *Owner Id* for the object. The object description names the owning module from the [MTE/SMTE](#) structures. See [.LM command](#) for related information.

##### 3. Process owner ([hptda](#))

Process owned objects are those allocated in the private arena of a running process and are either dynamic allocations or non-shared module segments, for example stack segments. The **hptda** of the process is used as the owner. The owner description names the process id and executable module of the owning process.

#### hmte

This names the [hmte](#) or [System Object Id](#) of the executable code that allocated the memory object or in the case of data segments of modules, which which they are associated.

#### sown

Semaphore owner id. This is the thread slot number that owns the memory management semaphore associated with this object. Memory management uses the address of the VMOB as the [block Id](#) to sleep on when the semaphore is held. This semaphore is used to serialise access to a VMOB structure. See [.PB command](#) for information on thread slots waiting on Block Ids.

#### cnt

Count of owners of the VMOB semaphore and wait flag.

The low order bit of **cnt** is used as a wait indicator. The high order 7 bits are a count of the number of times the owning thread has requested the VMOB semaphore without releasing it. See **sown** filed above for related

information.

xf

Extra flags.

The following flags are defined:

<i>name</i>	<i>bit mask</i>	<i>description</i>
VMOB_SLOCK_WAIT	0x01	Waiting on short term locks to clear
VMOB_LLOCK_WAIT	0x02	Waiting on long term locks to clear
VMOB_DISC_SEG	0x04	Object is part of a discardable seg
VMOB_HIGHMEM	0x08	Object was allocated via dh_vmalloc using the VMDHA_USEHIGHMEM flag

#### Notes:

The lock wait flags indicate that a thread is waiting for locked pages in the memory object to be unlocked, but not to resolve a conflicting lock request: that is indicated with the **OB\_LOCKWAIT** flag.

If a thread blocks waiting for long-term locks to clear then the address of the long-term lock count (VMOB + 0xd) is used as the Block-Id the thread blocks on. The thread blocks indefinitely.

If a thread blocks waiting for short-term locks to clear then the address of the short-term lock count (VMOB + 0xe) is used as the Block-Id the thread blocks on. The thread will block for up to 10 seconds. If after that time the short-term lock has not been cleared then an error is returned and under the debug kernel the following message is sent to the debug console:

**VMLOCK: Short term lock for > 10 secs: hob=hob**

See [.PB command](#) for information on thread slots waiting on Block Ids.

lt

Count of active long-term lock holders. A non-zero value indicates one or more pages of the memory object have been long-term locked, that is prevented from being paged out from physical storage. Long-term locks are expected to be held for a relatively long period of time, in the order of seconds. See [.MP command](#) for information on displaying physical storage status. See also [VM Lock Trace](#) Kernel Debugger facility.

st

Count of active short-term lock holders. A non-zero value indicates one or more pages of the memory object have been short-term locked, that is prevented from being paged out from physical storage. Short-term locks are expected to be held for a relatively short period of time, in the order of milli-seconds. See [.MP command](#) for information on displaying physical storage status. See also [VM Lock Trace](#) Kernel Debugger facility.

#### *description*

The object description appears to the right of the tabular display. It is combines the interpretation of **own** and **hmte** fields. The following forms are possible:

#### Process owned objects

These appear as:

**priv *pid* *process***

where:

*pid*

is the owning  
process id  
is the  
executable  
running the  
owning  
process

*process*

#### MTE Owned objects

These appear as:

**shared *module***

	where:	<i>module</i>	is the name of the module that contains the object ( <b>hob</b> ) displayed.
PTDA Pseudo-objects	These appear as:	<i>ptda pid process</i>	
	where:	<i>pid</i>	is the process id in which the object is located.
		<i>process</i>	is the executable running the owning process
MTE Pseudo-objects	These appear as:	<i>mte module</i>	
	where:	<i>module</i>	is the module name that corresponds to the <b>MTE</b> pointed to by the <b>va</b>
LDT occupying storage	This appears as:	<i>ldt pid process</i>	
	where:	<i>pid</i>	is the id of the process that owns the <b>LDT</b> related to the object
		<i>process</i>	is the executable running the owning process
Free objects	These appear as:	<i>free</i>	
System Owned Objects	These appear as:	<i>owner user</i>	
	where:	<i>owner</i>	is the <b>system object</b> name corresponding to the <b>own</b> field.
		<i>user</i>	is the <b>system object</b> name corresponding to the <b>hmte</b> field.

**Note:** From fix pack 29 of Warp 3.0 and GA Warp 4.0 the **hmte** of all objects with a system owner id is also formatted in the ***user*** part of the description.

-----

# .MP - Format Memory Page Frame Structures (PFs)



Display memory Page Frame Structures (PFs).

**Syntax:**

```
.MP
      B      F      frame
      I      laddr      Ln
      L
      R
```

**Parameters:**

- B** Display in-use (busy) Page Frame Structures in sequential order.  
**Note:** In-Use PFs are signified by the PF\_FREE flag being reset and not by the PF\_BUSY flag being set.
- F** Display free Page Frame Structures.
- I** Display idle Page Frame Structures. (This option was introduced at fix pack 29 of Warp V3 and base Warp V4.)
- L** Follow left (forward) chain pointer. This is only of relevance to Free and Idle Page Frame Structures since these are linked in a double linked chain.  
**Warning:** Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly. In particular the 2 high order digits of the frame number are truncated. Use this option advisedly! This is fixed from fix pack 29 for Warp V3 and base Warp V4. See [Free Page Frame Structures](#) or [Idle Page Frame Structures](#) for information on locating Idle and Free PF chains.

R

Follow right (backward) chain pointer. This is only of relevance to Free and Idle Page Frame Structures since these are linked in a double linked chain.

**Warning:** Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly. In particular the 2 high order digits of the frame number are truncated. Use this option advisedly! This is fixed from fix pack 29 for Warp V3 and base Warp V4. See [Free Page Frame Structures](#) or [Idle Page Frame Structures](#) for information on locating idle and free PF chains.

*laddr*

Specifies the linear address of a specific Page Frame Structure to be formatted.

An [address expression](#) may be specified.

*Ln*

Specifies the number of Page Frame Structures to display from the starting criterion.

*frame*

Specifies a physical storage page frame number. This will cause the Page Frame Structure corresponding to that frame to be displayed except for [UVIRT](#) storage. PFs corresponding to UVIRT storage are zeroed unless aliased by non-UVIRT storage. In the former case .MP will display then next non-UVIRT PF. In the latter it will display the aliasing non-UVIRT PF. See [DP command](#) for related information.

### Results & Notes:

Page Frame Structures are allocated in contiguous storage from the address given by global variable:

`_pft`

Output from the **.MP** command appears in one of three formats.

[In-use Page Frame Structure](#)  
[Idle Page Frame Structure.](#)  
[Free Page Frame Structure.](#)

For a description of the fields formatted by **.MP** select [.MP Output Field Descriptions](#)

For more examples using of the **.M** family of commands see: [Exploring Memory Management](#).

-----

## Free Page Frame Structures

```
ffdf509c Free:      BLink=0000f Flg=4 FLink=001da Blk=00001 Frame=0000d
ffdf50a8 Free:      BLink=001f2 Flg=4 FLink=0003f Blk=00001 Frame=0000e
ffdf50b4 Free:      BLink=001f1 Flg=4 FLink=0000d Blk=00001 Frame=0000f
ffdf50c0 Free:      BLink=001fe Flg=4 FLink=001f1 Blk=00000 Frame=00010
```

Free Page Frame Structures.

### Notes

- Free Page Frame Structures are chained in a double linked-list. The head of this list may be located as follows:
  - Locate list structure whose address is given by `_pgFreeList`
  - The first double-word of the list structure points to the pseudo-page frame structure that heads the free list.
  - The second double-word contains the pseudo-frame number of the pseudo-PF. N.B. This marks the end of the linked list only.
  - The backward pointer to the first true free PF is given by the 5 low order digits of the second double-word of the

pseudo-PF. This value may be used with the **.MP** command to format the first free **PF** provided it is not equal to the pseudo-frame number, which would imply no free **PFs** on the free list.

5. Subsequent free **PFs** are found by following the **BLink** until the pseudo-frame is encountered.
- The **Blk** field has a residual field and is of no direct relevance to free Page Frame Structures.

For a description of the fields formatted by **.MP** select [.MP Output Field Descriptions](#)

-----

## Idle Page Frame Structures

```
ffdfcdb8 Idle: pVP=ff1e6b9c Blink=01279 Flg=0 Flink=0004c Blk=0004a Frame=0127a
ffdfcdac Idle: pVP=ff1e6ba8 Blink=01261 Flg=0 Flink=0127a Blk=0004b Frame=01279
ffdfcc8c Idle: pVP=ff1e6c08 Blink=0125d Flg=0 Flink=01279 Blk=00066 Frame=01261
```

Idle Page Frame Structures.

### Notes

- Idle Page Frame Structures are chained in a double linked-list. The head of this list may be located as follows:
  1. Locate list structure whose address is given by **\_pgIdleList**
  2. The first double-word of the list structure points to the pseudo-page frame structure that heads the idle list.
  3. The second double-word contains the pseudo-frame number of the pseudo-PF. N.B. This marks the end of the linked list only.
  4. The backward pointer to the first true idle PF is given by the 5 low order digits of the second double-word of the pseudo-PF. This value may be used with the **.MP** command to format the first idle **PF** provided it is not equal to the pseudo-frame number, which would imply no idle **PFs** on the idle list.
  5. Subsequent idle **PFs** are found by following the **BLink** until the pseudo-frame is encountered.

For a description of the fields formatted by **.MP** select [.MP Output Field Descriptions](#)

-----

## In-use Page Frame Structures

```
ffdf5000 InUse: pVP=ff1df000 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00000
ffdf500c InUse: pVP=ff1df060 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00001
ffdf5018 InUse: pVP=ff1df06c RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00002
ffdf5024 InUse: pVP=ff1df078 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00003
```

In-Use Page Frame Structures.

For a description of the fields formatted by **.MP** select [.MP Output Field Descriptions](#)

-----



# .MP Output Field Descriptions

Output from **.MP** appears in one of the following forms:

```
ffdf500c InUse: pVP=ff1df060 RefCnt=0001 Flg=0 ll=00 sl=00 Blk=00000 Frame=00001
ffdfcdac Idle:  pVP=ff1e6ba8 Blink=01261 Flg=0 Flink=0127a Blk=0004b Frame=01279
ffdf50b4 Free:                                BLink=001f1 Flg=4 FLink=0000d Blk=00001 Frame=0000f
```

Each of the fields has the following meaning:

## ***address***

The linear address of the PF structure is given to the left of each display line.

## ***type***

The type of PF is displayed in the second column. Three types are possible: **Free Idle** and **InUse**.

## **pVP=**

The linear address of the associated [Virtual Page Structure](#). See [.MV Command](#) for information on displaying Virtual Page Structures.

## **RefCnt**

The number of [PTEs](#) that reference the frame of physical storage represented by this PF. A reference count greater than 1 indicates shared memory, some instances of which will be represented by [VMCOs](#) (see [.MC command](#)).

When a PTE is attached to an existing PF then the **Refcnt** is incremented.

When a page of memory is freed, the **Refcnt** is decremented. If it becomes zero the PF may be eligible for putting on the Idle list.

PFs corresponding to [UVIRT](#) storage are zeroed unless aliased by non-UVIRT storage. In either case no reference accounting is performed for UVIRT mappings.

## **Blink=**

The backward or right link to the previous Idle or Free PF.

## **Flink=**

The forward or left link to the next Idle or Free PF.

## **Flg=**

PF flags.

The following flags are defined:

<i>name</i>	<i>bit mask</i>	<i>description</i>
PF_FAST	0x1	frame is fast memory
PF_BUSY	0x2	frame is busy
PF_FREE	0x4	frame is free
PF_RES	0x8	reserved

**Notes:** **PF\_FAST** flag is set for some physical storage frames below 640K.

**PF\_BUSY** signifies that access to the PF is being serialised by the page frame manager. This is normally followed by setting the **VP\_BUSY** flag in the associated VP, if reset or setting the **VP\_WANTED** flag and waiting on the the [Block Id](#) of the VP address. Under the debug kernel the thread slot of the VP semaphore owner is saved in **vp\_semowner** (VP+0x0a) See [.PB command](#) for information on thread slots waiting on Block Ids.

## **ll=**

Count of number of long-term lock requests active against this page frame. This is incremented when a request to

lock a range of pages of a memory object is made. It is also, but rarely, set to 1 to isolate page frames that have caused trap 2 errors from which the system has recovered. See also [.MO command output](#) for flags relating to memory object locking.

**sl=**

Count of number of short-term lock requests active against this page frame. This is incremented when a request to lock a range of pages of a memory object is made. For related information, see [.MO command output](#) for flags relating to memory object locking.

**Blk=**

Specifies the swap disk frame, loader block number or diagnostic flag depending on the flag settings of the corresponding Virtual Page Structure pointed to by the **pVP=** field.

When **VP\_DF** is set and **VP\_DISCARDABLE** is reset then **Blk=** is the swap disk frame number that contains a copy of the page frame.

When **VP\_DISCARDABLE** is set and **VP\_RESIDENT** is reset then the **Blk=** field is the Loader block Id. Except for a special case noted below, this is a page index, starting from 1, into the objects of the module as an aggregated whole, with the size of each object rounded up to a page boundary. The special case occurs when the memory object that owes this page frame has an **hmte** set to [system object id 0xffc0](#), **Discard Owner**. When this occurs the following special block numbers may be used:

0x0ffe	System Infoseg
0x0ffd	Local Infoseg
0x0ffc	invalid LDT pages

When **VP\_DF** and **VP\_DISCARDABLE** are reset the **Blk=** usually indicates the last cross-linked swapper disk frame (unless its zero), however under the debug kernel negative values are used to indicate errors or instances where swapper frames have been freed because the corresponding [PTE](#) for the frame was found to be dirty. The following error indicators are possible:

-1	When also <b>Flg=9</b> then the physical frame caused a Trap 2 error, but the system was able to recover the data. The frame is isolated from further use by setting <b>Il=1</b> , <b>refcnt=1</b> and <b>PF_FREE</b> flags are reset and <b>pVP=pgVPBasePg</b>
-3	A page-in operation failed with ERROR_SWAP_IO_PROBLEMS
-4	A page_out operation failed with PGPO_FAILED
-5	A page_out operation failed with ERROR_SWAP_FILE_FULL

Otherwise disk frame reclamation is indicated by **Blk=** values of: -1, -2, -7, -9 and 0xffff0.

For related [VP](#) information, see [.MV command](#).

**Frame=**

This is the physical page frame number that this Page Frame Structure represents.

-----

## .MV - Format Memory Virtual Page Structures (VPs)



Display memory Virtual Page Structures ([VPs](#)).

### Syntax:

```
.MV
      B          F          vpid
      L          laddr      Ln
      R
```

### Parameters:

**B**

Display in-use (busy) Virtual Page Structures in sequential order.

**Note:** In-Use VPs are signified by a zero reference count and not by the The **VP\_BUSY** flag. See **Ref=** field in [.MV Output Field Description](#).

**F**

Display free Page Frame Structures.

**L**

Follow left (forward) chain pointer. This is only of relevance to Free Virtual Page Structures since these are linked in a double linked chain.

**Warning:** Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly and under some circumstances the debug kernel may hang. Use this option advisedly! From fix pack 29 for Warp V3 and base Warp V4 this has been fixed.

See [Free Virtual Page Structures](#) for information on locating Free VP chains.

**R**

Follow right (backward) chain pointer. This is only of relevance to Free Virtual Page Structures since these are linked in a double linked chain.

**Warning:** Both the Dump Formatter and the Kernel Debugger may fail to recognise the chain pointers correctly and under some circumstances the debug kernel may hang. (.MVFL will cause this effect). Use this option advisedly! From fix pack 29 for Warp V3 and base Warp V4 this has been fixed.

See [Free Virtual Page Structures](#) for information on locating Free VP chains.

***laddr***

Specifies the linear address of a specific Virtual Page Structure to be formatted.

An [address expression](#) may be specified.

***Ln***

Specifies the number of Virtual Page Structures to display from the starting criterion.

***vpid***

Specifies a VP Id. This is an index in to the table of Virtual Page Structures, which are located in contiguous virtual storage.

### **Results & Notes:**

Virtual Page Structures are allocated in contiguous storage from the address given by global variable:

**`_pgpVPBase`**

Output from the **.MV.** command appears in one of two formats.

[In-use Virtual Page Structure](#)  
[Free Virtual Page Structure.](#)

For a description of the fields formatted by **.MV** select [.MV Output Field Descriptions](#)

For more examples using of the **.M** family of commands see: [Exploring Memory Management](#).

---

## Free Virtual Page Structures

```
VPI=0d3e pVP=ff1e8ee8 free FLink=00000000 BLink=fff13280
VPI=0d3f pVP=ff1e8ef4 free FLink=ff1e9fec BLink=ff1e8cf0
VPI=0d40 pVP=ff1e8f00 free FLink=ff1e9fec BLink=ff1e8cf0
VPI=0d41 pVP=ff1e8f0c free FLink=00001000 BLink=02450030
VPI=0d42 pVP=ff1e8f18 free FLink=00000000 BLink=ff1e8f00
```

Free Virtual Page Structures.

#### Notes

- Free Page Frame Structures are grouped in bundles that are chained in a circular double link list. Each bundle comprises contiguous free VPs in the VP array. The chain pointers are only used by the head and tail of each bundle as follows:

- For bundles of greater than one VP:
  1. **Blink** of the head points to the tail
  2. **Flink** of the head points to the head of the next bundle
  3. **Blink** of the tail points to the head of the previous bundle
  4. **Flink** of the tail is set to zero
- For single VP bundles:
  1. **Blink** points to the head of the previous bundle
  2. **Flink** points to the head of the next bundle

The free VP chain is headed by a pseudo-VP whose **Blink** points to the head of the first true free bundle and whose **Flink** points to the last VP in the VP array. The pseudo-VP is located at global symbol:

`_pgVPHead`

- Unless a free VP is the head or tail of a bundle then its **Flink** and **Blink** will retain values from its previous use. In particular it may be possible to glean information about a previous allocation as the **Flink** field overlays the **Flg** and **Block** fields and the **Blink** field overlays the **HobPg** and **Hob** fields of an In-use VP. In the example above VPI d41 was probably allocated to page 30 of [hob](#) 245. Using the following [.MOC command](#) might reveal who the owner was and who allocated this storage as follows:

`.MOC 245`

For a description of the fields formatted by `.MV` select [.MV Output Field Descriptions](#)

For more examples using of the `.M` family of commands see: [Exploring Memory Management](#).

-----

## In-use Virtual Page Structures

```
VPI=0000 pVP=ff1df000 Res Frame=0000 Flg=410 HobPg=0000 Hob=ff77 Ref=001 Own=000
VPI=0001 pVP=ff1df00c Res Block=0000 Flg=c00 HobPg=0000 Hob=ff6c Ref=042 Own=000
VPI=0002 pVP=ff1df018 Res Frame=0bc5 Flg=410 HobPg=0000 Hob=0001 Ref=001 Own=000
VPI=0003 pVP=ff1df024 Res Frame=0bc4 Flg=410 HobPg=027a Hob=0022 Ref=001 Own=000
```

Free Virtual Page Structures.

For a description of the fields formatted by `.MV` select [.MV Output Field Descriptions](#)

-----

## .MV Output Field Descriptions

Output from `.MV` appears in one of is of the following forms:

```
VPI=0000 pVP=ff1df000 Res Frame=0000 Flg=410 HobPg=0000 Hob=ff77 Ref=001 Own=000
VPI=0001 pVP=ff1df00c Res Block=0000 Flg=c00 HobPg=0000 Hob=ff6c Ref=042 Own=000
```

VPI=0d40 pVP=ff1e8f00 free FLink=ff1e9fec BLink=ff1e8cf0

Each of the fields has the following meaning:

**VPI=**

The VP index into the array of VPs.

**pVP=**

The linear address of the VP.

***status***

The status of the VP interpreted from the **Flg** field. The following values may appear:

SOW	Swap on Write flag (VP_SOW set)
Res	Page is resident (VP_RESIDENT set)
Dsc	Page is discardable (VP_DISCARDABLE set)
Swp	Page is swappable (VP_DISCARDABLE reset)
free	VP is free (vp_refcount=0)

**Block=nnnn**

The cross-linked loader block number or swapper disk frame. This implies the virtual page is not attached to a [PF](#). If it is:

**discardable** then it is linked to a loader block id,

**swappable** then it is linked to a swapper disk frame.

When the page is swappable (VP\_DISCARDABLE reset) and does not have a disk frame (VP\_DF reset) then the following special **Block** values may be used:

0	Allocate PF on demand
1	Allocate on demand zero-fill page
2	page is in a broken disk frame

**Frame=nnn**

The virtual page is linked to [PF nnnn](#). Refer to the [.MP command](#) for displaying information about the related page frame.

**Flink=**

Forward link of a free VP. This is only of relevance to the VP at the head of a bundle of free VPs. See [Free Virtual Page Structures](#) for information on how free VPs are linked.

**Blink=**

Backward link of a free VP. This is only of relevance to the VP at the head and tail of a bundle of free VPs. See [Free Virtual Page Structures](#) for information on how free VPs are linked.

**Flg=**

VP flags.

The following flags are defined:

<i>name</i>	<i>bit mask</i>	<i>description</i>
VP_BUSY	0x001	page semaphore taken
VP_WANTED	0x002	page semaphore requested
VP_CACHE	0x004	search page cache for pf
VP_PFIDLE	0x008	cross linked to idle pf
VP_PF	0x010	cross linked to pf
VP_DF	0x020	has swap file disk frame
VP_DIRTY	0x040	contents written to - from pte
VP_SHDIRTY	0x080	shadow dirty bit (for VDMs)
VP_SOW	0x100	change to swappable on write

VP_PRIVATIZED	0x200	vp privatized
VP_RESIDENT	0x400	cannot be moved - value from pte
VP_DISCARDABLE	0x800	1 = discardable, 0 = swappable

#### Notes:

**PF\_BUSY** signifies that access to the VP is being serialised by the page frame manager.

**VP\_WANTED** signifies that a thread is waiting to mark the VP busy. The thread will wait on a [Block Id](#) of the VP address. Under the debug kernel the thread slot of the VP semaphore owner is saved in **vp\_semowner** (VP+0x0a) see **Own=** field of the .MV display. See [.PB command](#) for information on thread slots waiting on Block Ids.

#### HobPg=

The relative page number of the memory object that this VP is assigned to. See **Hob=** field below.

#### Hob=

The [hob](#) of the memory object to which this page is assigned.

**Note:** Use

```
.MOC hob
```

to obtain the virtual address and owner information relating to this VP. See [.MO command](#) for more information.

#### Ref=

The number of memory objects sharing this page of data. A reference count greater than 1 indicates shared memory, some instances of which will be represented by [VMCOs](#) (see [.MC command](#)) and others by aliases (see [.ML command](#)).

The reference count is incremented and decremented according to usage. When the count becomes zero the VP is no longer in use and any committed physical storage or swapper storage may become eligible for freeing.

[UVIRT](#) storage is not represented by VPs thus reference accounting is not performed.

#### Own=

The thread slot number of the current owner of the VP semaphore. This field is only used in the debug kernel and will only have significance if the **VP\_BUSY** or **WP\_WANTED** flags are set. See [.PB command](#) for information on thread slots waiting on Block Ids.

-----

## .N - Display Dump Information Summary



Display information saved by from the operating system when the stand alone dump procedure was initiated.

#### Syntax:

```
.N
```

#### Parameters:

*none*

#### Results & Notes:

**.N** command displays information saved when the kernel routine **RASRST** is entered at sand-alone dump initiation.

**.N** displays the following information:

```
gdtr_lim: 1FFF
gdtr_base: 7C3E5000
ldtr_lim: 03FF
ldtr_base: FFE00150
ldtr_reg: 0028
lo_data_sel: 0400
hi_data_sel: 0400
trace_buf_addr: 0B490400
sys_anchor_sel: 0070
arena_base: FEB1F020
max_threads: 0101
phys_page_dir: 001D6000
vm_object_ptr: FEC80020
StartInit_Data: 00000140
dcm_ote_start: FFF0A92F
CurProcPid: 000D
TaskData: 0B5C0400
FirstPacket: 158A
LastPacket: 04C0
SysSemDataTable: 53A60400
GDT_Buffers: 00A80138
PapTCBPtrs: 0B6B0400
callerSS: 00E8
callerESP: 00000FCC
savePage: 00241467
```

Each of the items displayed has the following significance:

<b>gdtr_lim</b>	The current GDTR register limit value.
<b>gdtr_base</b>	The current GDTR register base address.
<b>ldtr_lim</b>	The current LDTR selector limit.
<b>ldtr_base</b>	The current LDTR selector base address.
<b>ldtr_reg</b>	The current LDT selector.
<b>lo_data_sel</b>	Selector for <b>DOSGROUP</b> segment.
<b>hi_data_sel</b>	Selector for <b>DOSHIGHDATA</b> segment.
<b>trace_buf_addr</b>	<b>Offset:selector</b> address of <b>ras_stda_addr</b> , the selector for the system trace buffer.
<b>sys_anchor_sel</b>	Selector for the <b>SAS</b> .
<b>sarena_base</b>	Value of <b>_parvmOne</b> , the pointer to the first <b>VMAR</b> .
<b>max_threads</b>	Maximum <b>Thread Slot Number</b> .
<b>phys_page_dir</b>	Value of cr3 register (that is, the physical address of the page directory table).
<b>vm_object_ptr</b>	Value of <b>_pobvmOne</b> , the pointer to the first <b>VMOB</b>

<b>StartInit_data</b>	Value of <b>_StartINITData</b> .
<b>dcm_ote_start</b>	Address of <b>DOSCALLS.DLL</b> <a href="#">OTE</a>
<b>CurrProcPid</b>	Current process ID.
<b>TaskData</b>	<b>Offset:selector</b> address of scheduler global data.
<b>FirstPacket</b>	First word of the first device driver strategy 2 request packet in packet pool.
<b>LastPacket</b>	First word of the last device driver strategy 2 request packet in packet pool.
<b>SysSemDataTable</b>	<b>Offset:selector</b> address of <b>SysSemDataTable</b> .
<b>GDT_Buffers</b>	GDT selector for buffer segment. The low order word of this field should be ignored.
<b>PapTCBPtrs</b>	<b>Offset:selector</b> address of <b>papTCBPtrs</b> . The word value at this label is an offset from the <b>DOSGROUP</b> selector (400) to the thread slot table.
<b>callerSS</b>	The <b>SS</b> selector on entry to <b>RASRST</b> , the stand alone system dump entry point within the kernel.  <b>Note:</b> If this dump was taken in interrupt mode the <b>SS</b> selector will be <b>E8</b> . Further more, if that last device driver to use the interrupt stack is <b>KDB\$</b> then it is possible the the dump process was initiated with the <b>Ctrl-Alt-Numlock-Numlock</b> sequence. If the dump was taken in kernel mode then the <b>SS</b> selector will probably be <b>30</b> and the dump will have been initiated because of a trap or call to <b>DosForceSystemDump</b> API.
<b>callerESS</b>	The <b>ES</b> register on entry tp <b>RASRST</b> , the stand alone system dump entry point within the kernel.
<b>savePage</b>	Page directory entry 0. This data is overwritten by the dump process.

-----

## .O - Override default behaviour



Override the default system behaviour for handling certain events.

### Syntax:

```
.O      Q      n
        N
        F
```

### Parameters:

- Q**                      Query override settings in effect.
- N**                      Activate (turn oN) overrides for event **n**.



**F** Deactivate (turn off) overrides for event *n*.

*n* Specifies the override event number. Currently only one override event is defined, which is event number 1. When active this forces breakpoints to be serviced by the Kernel Debugger instead of the **DosDebug** or **DosPTrace** APIs.

This only affects debugging situations where a debugging application is active under the Kernel Debugger.

**Results & Notes:**

This command was introduced with fix pack 29 for Warp 3.0 and fix pack 1 for Warp 4.0.

Prior to the introduction of this command, the only way to override **DosDebug** and **DosPTrace** was to zero to address of the Debug Control Block ((**ptda\_pdc** + 0x38)) in the **PTDA** of the debuggee process.

**.OQ** displays information about overrides in a tabular form as shown in the following example:

```
#.OF 1
DosDebugAPI Override is now OFF.
#.OQ
Override(s) Query Information
# | Override      | Status
-----
1 | DosDebugAPI    | OVERRIDE OFF
-----

#.ON 1
DosDebugAPI Override is now ON.
#.OQ
Override(s) Query Information
# | Override      | Status
-----
1 | DosDebugAPI    | OVERRIDE ON
-----
```

**.P - Display Process Status**



Display **process** and **thread** status information from the Per Task Data Area (**PTDA**), Thread Control Block (**TCB**) and Thread Swappable Data (**TSD**).

**Syntax:**

```
.P
#
*
slot
```

**Parameters:**

*slot* Display process status for **thread slot** *slot*.

The following short-hand may be used for the slot number:

\* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

**\_\_TaskNumber**

# The debugger default thread slot. This defaults to the current slot unless overridden by the [.S command](#).

If no slot number is given then all thread slots are displayed, grouped by process.

### Results & Notes:

The **.P command** locates a thread's TCB from either the thread slot table, the linear address of which is given by global variable:

`_papTCBSlots`

or by traversing the process tree using **TCBpTCBNext** (TCB +0x14), **TCBpPTDA** (TCB +0x08) and **ptda\_pTCBHead** (PTDA + 0x20) fields. Output from the **.P** command appears in tabular form as follows:

Slot	Pid	Ppid	Csid	Ord	Sta	Pri	pTSD	pPTDA	pTCB	Disp	SG	Name
0001	0001	0000	0000	0001	blk	0100	ffe3a000	ffe3c7d4	ffe3c61c	1eb4	00	*ager
0002	0001	0000	0000	0002	blk	0200	7b7ca000	ffe3c7d4	7b9c8020	1f3c	00	*tsd
0003	0001	0000	0000	0003	blk	0200	7b7cc000	ffe3c7d4	7b9c81d8	1f50	00	*ctxh
0004	0001	0000	0000	0004	blk	081f	7b7ce000	ffe3c7d4	7b9c8390	1f48	00	*kdb
0005	0001	0000	0000	0005	blk	0800	7b7d0000	ffe3c7d4	7b9c8548	1f20	00	*lazyw
0006	0001	0000	0000	0006	blk	0800	7b7d2000	ffe3c7d4	7b9c8700	1f3c	00	*asynchr
*0008	0002	0001	0002	0001	blk	0500	7b7d6000	7b9e4020	7b9c8a70	1eb8	01	pmsshell
000a#	0002	0001	0002	0002	blk	0800	7b7da000	7b9e4020	7b9c8de0	1ed4	01	pmsshell

Each of the fields has the following meaning:

#### slot

The unique (hexadecimal) index in to the thread slot table of all threads.

This value may be flagged with:

\* to the left to signify the current (or last) dispatched thread.  
# to the right to signify the Kernel Debugger or Dump Formatter default thread slot.

**Slot** may be found in the **TCBNumber** (TCB + 0x2) field of the **TCB**

#### Pid

The [process id](#) this thread slot is assigned to.

#### Ppid

The parent [process id](#) that created this thread. A value of zero signifies a detached process.

#### Csid

The [command subtree id](#).

This is normally the same value as the **Pid**. When the parent process dies any orphaned children are adopted by their grandparent by making **Ppid** equal to the grandparent's **Pid**. Each orphan inherits the **Csid** of its dying parent. This mechanism ensures that orphaned PTDA's are not retained for returning termination information to their parent (via DosWaitChild).

**Csid** is taken from the **Csid** (PTDA +0x4be (H/R: +0x4b6)) field of the **PTDA**

#### Ord

The thread ordinal for this thread slot. This is the unique [thread id](#) assigned to the thread within the process to which it belongs.

**Ord** is taken from the **TCBOrdinal** (TCB+ 0x0) field of the **TCB**

#### Sta

The thread's ascending [scheduler](#) state taken from the **TCBState** (TCB +0x161) field.

Except when a state transition is progress this is the same as the current state of the thread (see **Qst** field of the [.PQ command](#).)

The following states are possible:

<i>abrv</i>	<i>Sta</i>	<i>TCBState description</i>
		<i>value</i>

---	STATE_VOID	0	Uninitialized or Dead thread
rdy	STATE_READY	1	Thread ready to run
blk	STATE_BLOCKED	2	Blocked on a <a href="#">Block Id</a>
sus	STATE_SUSPENDED	3	*** Not in Use ***
crt	STATE_CRITSEC	4	Blocked by another CritSec thread (after attempting to run)
run	STATE_RUNNING	5	Thread currently running
bst	STATE_READYBOOST	6	Ready, but apply an IO boost after swapping in a TSD
tsd	STATE_TSD	7	Thread waiting for the TSD daemon to page in the TSD.
dly	STATE_DELAYED	8	Delayed TKWakeup (Almost Ready)
frz	STATE_FROZEN	9	Frozen Thread via DosSuspendThread, DosCreateThread, DosExecPgm or DosSystemService
gsk	STATE_GETSTACK	10	TSD daemon is waiting for the page manager to page in a TSD
bad	STATE_BADSTACK	11	TSD failed to page in

#### Notes:

The scheduler manages threads on queues by priority and state. See [.PQ command](#) for displaying scheduler queues.

The scheduler uses a [finite state machine](#) to manipulate thread queues. **TCBQState** and **TCBState** are the state transition drivers. They hold a thread's current and desired state. Except during a state transition current and desired state will be identical.

**STATE\_RUNNING** is set when the next potential runner has been selected. The running thread's context is then switched and various dispatcher flags checked before finally giving control to user code. It is therefore possible for the running state to be set and for the user code not to run.

**STATE\_READYBOOST** is a modified ready state and never becomes the current state, instead a priority boost is applied and the state becomes **STATE\_READY**

**STATE\_CRITSEC** This state applies to non-critical section threads only. It is only set when a critical section thread within the same process has given up the processor, while still in critical section, and another thread in the same process is selected to run. If this thread is thread 1 of the process and there are pending signals to process, the thread's signal handler will be dispatched. When there are no more pending signals or this thread is not thread 1 then **STATE\_CRITSEC** will be set.

**STATE\_FROZEN** is normally only seen when an application uses the DosSuspendThread API or creates a thread (or process) that is initially suspended. DosSystemService is used by the session manage to freeze all threads of a process in one system call.

Many states are transient accordingly the persistent appearance of a particular state might indicate a problem of the following nature:

<b>rdy</b>	Many ready threads might indicate contention for processor time. Tends to indicate the existence of a higher priority CPU bound thread.
<b>run</b>	Under the Dump Formatter this would indicate a trapped or processor bound thread. Under the Kernel Debugger a processor bound thread.
<b>blk</b>	All threads blocked could indicate no-work or a

deadlock. Under Dump Formatter this would imply a manually invoked dump using Ctrl-Alt-Numlock-Numlock or use of the DosForceSystemDump API.

---	The void state is rarely seen. Under Dump Formatter this probably indicates an incorrect version of the Dump Formatter for the system dumped.
crt	Another thread in the same process is in critical section and is either blocking without exiting critical section or is processor bound.
dly	Another thread is processor bound.
frz	A deadlock, loop or no-work for the freezing thread
sus	Is not used, so probably indicates a mismatch between the Dump Formatter and dump.
tsd	Physical storage overcommitted. Swapper very large. System may be <a href="#">thrashing</a> .
bst	Physical storage overcommitted. Swapper very large. Very busy processor bound system System may be <a href="#">thrashing</a> .
gsk	Physical storage overcommitted. Swapper very large. Very busy processor bound system System may be <a href="#">thrashing</a> .
bad	Excessive swapper. System may die. Physical storage overcommitted. Very busy processor bound system.

## Pri

Thread priority (word length field) in **TCBPriority** (**TCB** +0x168).

This is the current priority calculated by the scheduler based upon priority class (**TCBPriClass** (**TCB** +0x164)), priority class level (**TCBPrilevel** (**TCB** +0x165)) and priority boosts (**TCBPriClassMod** (**TCB** +0x166)).

The following priority classes are defined:

<i>class</i>	<i>value</i>	<i>description</i>
CLASS_IDLE_TIME	0x01	Idle-Time class
CLASS_REGULAR	0x02	Regular class
CLASS_TIME_CRITICAL	0x03	Time-Critical class
CLASS_SERVER	0x04	Client-Server Server class

The following priority boosts (class modifiers) are defined:

<i>Boost</i>	<i>Value</i>	<i>Description</i>
CLASSMOD_KEYBOARD	0x04	Keyboard boost
CLASSMOD_STARVED	0x08	Starvation boost
CLASSMOD_DEVICE	0x10	Device I/O Done Boost
CLASSMOD_FOREGROUND	0x20	Foreground boost
CLASSMOD_WINDOW	0x40	Window Boost

CLASSMOD_VDM_INTERRUPT	0x80	VDM simulated interrupt boost
------------------------	------	----------------------------------

**Note:**

**CLASSMOD\_KEYBOARD** has no effect on **CLASS\_SERVER**

The priority level is a value between 0x0 and 0x1f.

Priority class and modifier values are logically ORed to form an index into the priority class translation table, which is located at global symbol:

`_schPriClassTbl`

The resulting value is logically ORed with the priority level. The final value is subject to the minimum thread priority (**TCBPRIORITYMin** (**TCB** +016a)).

Priority boosts do not affect the priority of idle and time-critical threads.

Priority level has little or no effect on the priority of boosted regular and server class threads. threads

**pTSD**

Linear address of the **TSD** control block associated with this thread this thread taken from the **TCBpTSD** (**TCB** +0x0c).

**Note:**

The TSD contains the ring 0 stack for the associated thread. For the current thread this is addressable from selector 30 however the base address of selector 30 is entirely different from **TCBpTSD**. This is because the two addresses are aliased using two **PTEs** to pin the same physical frame. This device allows the TSD for be accessed out-of-context by the system, at the same time protecting system code from erroneous stack references.

**pPTDA**

Linear address of the control block representing the process to which this thread belongs. The address is taken from **TCBpPTDA** (**TCB** +0x08).

**pTCB**

Linear address of the **TCB** control block which represents the thread.

**Note:**

The output from **.P** is ordered by process and child process. TCBs are initially located from the thread table then the chain pointer **TCBpTCBNext** (**TCB** +0x14) is used to locate the remaining threads of a process.

Under the Dump Formatter **.P** will occasionally miss a thread because of the non-sequential manner in which the thread table slots are re-used. To ensure all active threads are displayed use **.PU**, or **.PQ** commands.

**Disp**

The displacement into the **TSD** for the current thread that the dispatcher will use for its ESP after having switched back to this thread's context.

This value is calculated from **TSDKernelESP**, therefore requires the **TSD** to be present. If the **TSD** is not present then a blank value is given. The **TSD** may be forced present under the Kernel Debugger by use of the **.I command**.

**SG**

**Screen Group ID** currently assigned to this process.

The Screen Groups ID is taken from the console locus structure (**Cons\_Loc** +0x2) embedded in the **PTDA** ((**PTDA** +0x526 (H/R: +0x51e))).

**Name**

The name of the primary executable running in this process.

Except for process 1 and Dos Virtual Machines the name is obtained from the **hmt** stored in **ptda\_module** (**PTDA** +0x5a6 (H/R: +0x59e)). If the **SMTE** is paged in then the name is taken from the file name pointed to by **smte\_path** otherwise it is taken from **mte\_module** and prefixed with an ! point. See **.LM command** for information on formatting loader control blocks.

Process 1 comprises internal threads, that is threads which run in the kernel and are not separately loaded modules. **ptda\_module** is zero for this process so the Dump Formatter and Kernel Debugger translate the Tids for Pid 1 as follows:

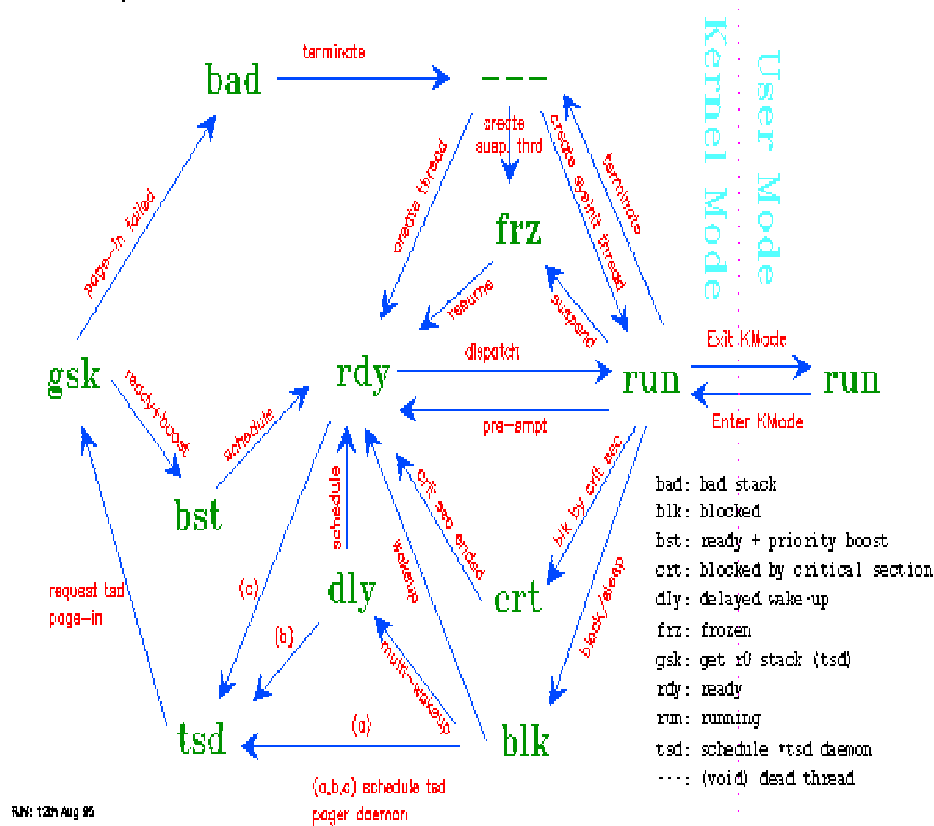
<i>Tid</i>	<i>name</i>	<i>description</i>
1	*ager	Ager thread used for compressing the Swap File.
2	*tsd	Scheduler's Daemon Thread used to page in TSDs
3	*ctxh	Default Global Context Hook dispatching thread.
4	*kdb	Kernel Debugger Daemon thread used to process page-in requests from the <a href="#">.ID</a> command
5	*lazyw	File system cache lazy writer thread.
6	*asynr	File system asynchronous read ahead thread.
7	*sysinit	System initialisation thread.
8-n		Other transient internal threads associated with system initialisation have a blank name.

Virtual DOS Machines run the [DEM](#) component of OS/2 to provide DOS emulation. DOS programs are loaded by the DEM and not known to the (OS/2) loader. Thus ptda\_module is zero and the Kernel Debugger and Dump Formatter use the name **\*vdm** to indicate a VDM. The [PSP](#) of the first loaded DOS program in a process may be located from **CurrentPDB (PTDA +0x2ea)**, which contains its segment address. The preceding [paragraph](#) contains the DOS arena record, the last 8 bytes of which contains the DOS program name currently executing.

-----

## Scheduler Finite State Machine

## OS/2 Scheduler Finite State Machine



## .PB - Display Blocked Thread Information



Display information about all blocked [threads](#).

### Syntax:

```
.PB
```

```
#
*
slot
```

### Parameters:

*slot*

Display user information for [thread slot](#) *slot*.

The following short-hand may be used for the slot number:

\*

The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

`_TaskNumber`

# The debugger default thread slot. This defaults to the current slot unless overridden by the [.S command](#).

If no slot number is given then all thread slots are displayed in slot number order.

### Results & Notes:

The **.PB command** locates each thread's [TCB](#) from the thread slot table, the linear address of which is given by global variable:

`_papTCBSlots`

or by traversing the process tree using **TCBpTCBNext** (**TCB** + 0x14), **TCBpPTDA** (**TCB** + 0x08) and **ptda\_pTCBHead** (**PTDA** + 0x20) fields.

Output is displayed only if a thread is blocked on a [BlockId](#). It appears in tabular form as follows:

Slot	Sta	BlockID	Name	Type	Addr	Symbol
0001	blk	fff11050	*ager			
0002	blk	fff74f59	*tsd			_tkTSDDaemon
0003	blk	fff43c78	*ctxh			_kmCTXHDaemon
0004	blk	fff7545a	*kdb			_tkKDBDaemon
0005	blk	fff02dfc	*lazyw			_semLW
0006	blk	fff111d4	*asynchr			_AsyncReadSem
0008	blk	fffe000e	pmsHELL	RamSem	074b:06d6	
000a	blk	ffca0002	pmsHELL			
000b	blk	fffd000b	pmsHELL	MuxWait		
000c	blk	fffd000c	pmsHELL	MuxWait		
000d	blk	04000df0	pmsHELL	DosSem	0400:0df0	CtrlNumLkQ
0007	blk	fe750a10	pmsHELL	Sem32	8001 0019	vhevLazyWrite
0010	blk	fe728dcc	pmsHELL	Sem32	8001 0001	SrvReq
0011	blk	fffe0006	pmsHELL	RamSem	d0c7:0020	
0012	blk	fffd0012	pmsHELL	MuxWait		
*0013	blk	fffe0007	pmsHELL	RamSem	d09f:0bc0	memory_pool + 127
0014#	blk	fffe0008	pmsHELL	RamSem	d09f:0bc8	memory_pool + 12f

Each of the fields has the following meaning:

#### slot

The unique (hexadecimal) index in to the thread slot table of all threads.

This value may be flagged with:

- \* to the left to signify the last dispatched thread.
- # to the right to signify the Kernel Debugger or Dump Formatter default thread slot.

**Slot** may be found in the **TCBNumber** (**TCB** + 0x2) field of the **TCB**

#### Sta

The ascending or desired state of the thread. This should always appear as **blk** for the **.PB** command, however Dump Formatter does not check the thread state so formats all threads. Those whose state is not **blk** should be ignored. From fix pack 29 for Warp V3 and base Warp V4 this has been fixed so that only blocked threads are formatted. See [Scheduler Finite State Machine](#) and [.PQ command](#) for more information on thread states.

#### BlockID

The token used by **TKSleep** and **TKWapeUp** to sleep and wake a thread on an event.

The **BlockId** is taken from **TCBSleepID** (**TCB** + 0x18c).

The **BlockID** is a conventional value. A number of conventions are used by various system components. Usually the **BlockID** is constructed so to be unique across all conventions. Frequently it will refer to the address of an associated resource, such as a system control block, or a field within a control block. See discussion of the **Type** field below for more information on interpreting **BlockIds**.

#### Name

The name of the primary executable running in this process.

See **name** field description of the [.P command](#) for further information.

#### Type



Interpretation of the use of the **BlockID** in conjunction with double word **TCB\_SemInfo** (**TCB** + 0x14c) and double word **TCB\_SemDebugAddr** (**TCB** + 0x150).

The following **Types** are recognised by the Dump Formatter and Kernel Debugger:

#### RamSem

The thread is waiting on a **RamSem** or **FastSafeRamSem**.

The high word of the **BlockID** is **0xfffe**, the low word is the **RamSemID** taken from the [RamSem structure](#).

The **Addr** field is taken from **TCB\_SemInfo**. This is a *selector:offset* address of the **RAMSEM**. The **RamSem** may be imbedded within a [FastSafeRamSem](#) or a [PMFastSafeRamSem](#).

The **Symbol** displayed is either that of the **TCB\_SemDebugAddr** or if -1, the **RamSem** address. See [LN command](#) for information on displaying symbols.

#### MuxWait

The thread is waiting on multiple events.

The high word of the **BlockID** is **0xffffd**, the low word is the **slot** of the waiting thread.

**TCB\_SemInfo** and **TCB\_SemDebugAddr** are not used with a **MuxWait BlockID**.

To locate the semaphores that comprise a given **MuxWait** proceed as follows:

- 1                    Locate the **MuxWait** table at symbol **MuxTable**. Display this using **DB** for convenience. This table comprises 9 byte entries whose format is given by the [MuxTableEntry](#) structure.
- 2                    Scan the **MuxTable** for entries that have this thread's slot number (+0x2 into each entry).
- 3                    Of those entries, select those with non-zero **MuxType** (+4 into each entry).
- 4                    Choose one of the following:
  - For type 1 (SysSem)

the last double word is the linear address of a system semaphore structure. Use the technique described below under **SysSem** for interpreting the **SysSem**.
  - For type 2 (RamSem)

the last double word of the entry contains the **RamSem** handle, the high word is the [hob](#) of the memory object containing the **RamSem**. The low word is the offset into the object where the **RamSem** is located. Use the technique described above under **RamSem** for interpreting the **RamSem**
  - For type 3 (Physical RamSem)

the last double word is the physical address of the **RamSem**.
  - For type 4 (32-bit event sem)

the last double word is the physical address of a 32-bit event semaphore. See **Sem32** below.

See [Mux Semaphore](#) example debugging log for an explicit example of using this technique.

**Addr** and **Symbol** fields are blank.

## ReqPkt

The thread is waiting for an I/O request packet to complete.

The **BlockID** is the *Selector:Offset* address of the request packet. The Selector is the **DOSGROUP** kernel selector and should be selector 400.

The address should lie between addresses at global symbols: **FirstPacket** and **LastPacket**.

See the Physical Device Driver Reference manual for information on device driver request packets.

**Addr** and **Symbol** fields are blank.

## SysSem

The thread is waiting on a system semaphore.

The **BlockID** is the *Selector:Offset* address of the [SysSemTblStruc](#) structure. The Selector is the **DOSGROUP** kernel selector and should be selector 400.

The address should lie within the System Semaphore Data Table, located at symbol **SysSemDataTable** for length 256\*6 bytes.

Offset +0 of each table entry contains the owner's thread number.

The name associated with the semaphore may be located as follows:

- 1                      Locate the **SysSem RMP** segment by displaying double word at symbol **SysSemRMPHdl**. The high word is the selector for the semaphore RMP.
- 2                      Display the System Semaphore RMP using **DB**. The first 0x14 bytes is the RMP header. The remainder comprises variable length records. The first word of each record is its length and therefore the relative offset to the beginning of the following record. Offset 2 of each record is the semaphore data table selector offset.
- 3                      Scan the RMP looking for an offset that matches the low word of the **BlockID**. When found the remaining bytes of the RMP record is the semaphore name (with the top two bytes overlaid by the semaphore offset).

See [System Semaphore](#) example debugging log for an explicit example of using this technique.

The **Addr** and **Symbol** fields are blank.

## DosSem

The thread is waiting on an internal **RamSem** or blocking on an address in the kernel's **DOSGROUP** segment.

The **BlockID** is the *selector:offset* of the **DosSem**. The Selector is the **DOSGROUP** kernel selector and should be selector 400. The offset does not lie in the System Semaphore Data Table or the I/O Request Packet Table.

**Addr** is the **BlockID** formatted as *selector:offset*.

The **Symbol** displayed is either that of the **TCB\_SemDebugAddr** or if -1, the **DosSem** address. See [LN command](#) for information on displaying symbols.

## Sem32

The thread is waiting on a 32-bit semaphore.

The **BlockID** is the address of the [32-bit Semaphore structure](#).

**TCB\_SemInfo** contains the semaphore handle. This is of the form:

<b>8001nnnn</b>	Shared 32-bit semaphore. <i>nnnn</i> is the (double-word) index into the shared semaphore table located at symbol <b>_pShSemTbl</b> . Each entry is an address of the corresponding 32-bit
-----------------	--

	semaphore structure. (that is, the <b>Sem32 BlockID</b> ).
<b>0001nnnn</b>	Private 32-bit semaphore. <i>nnnn</i> is the (double-word) index into the private semaphore table located at <b>pPrSemTbl</b> ( <b>PTDA</b> +0x4cc (H/R: +0x4c4)). Each entry is an address of the corresponding 32-bit semaphore structure. (that is, the <b>Sem32 BlockID</b> ).
	<b>Addr</b> field is the semaphore handle formatted as two words.
	The <b>Symbol</b> displayed is either that of the <b>TCB_SemDebugAddr</b> or if -1, the <b>Sem32</b> address. See <a href="#">LN command</a> for information on displaying symbols.
	Use the <a href="#">.D SEM32 comand</a> with the <b>BlockID</b> to format the 32-bit semaphore.
<b>Buffer</b>	<p>The thread is waiting for a file system buffer.</p> <p>The <b>BlockId</b> is the <i>selector:offset</i> address of the buffer. The high word is the buffer selector and should be <b>a8</b>.</p> <p>The <b>Addr</b> and <b>Symbol</b> fields are blank.</p>
<b>SFT</b>	<p>The thread is waiting for a <a href="#">SFT</a> entry.</p> <p>The <b>BlockId</b> is the <i>selector:offset</i> address of the SFT. The high word is the buffer selector and should be one that is listed in the <b>SFT</b> table pointed to by <b>c0:0</b>.</p> <p>The <b>Addr</b> and <b>Symbol</b> fields are blank.</p>
<b>ChildWait</b>	<p>The thread is waiting in DosWaitChild for a child process to terminate.</p> <p>The high word of the <b>BlockID</b> is the <b>ptda_Pid</b> offset from selector 30 (0xffca).</p> <p>The low word of the <b>BlockID</b> is the <a href="#">Pid</a> to which this thread belongs.</p>
<i>blank type</i>	<p>The thread is waiting on a <b>BlockId</b> that the Kernel Debugger and Dump Formatter have not been able to identify.</p> <p><b>Addr</b> field is blank.</p> <p>The <b>Symbol</b> displayed is either that of the <b>TCB_SemDebugAddr</b> or if -1, the <b>Sem32</b> address. See <a href="#">LN command</a> for information on displaying symbols.</p>

#### Notes:

The **BlockID** interpretation is not exact. A device driver, for example, could call **DevHlp\_ProcBlock** using a value for **BlockID** that conflicts with another convention.

Under the Debug kernel only, **TCB\_SemDebugAddr** is used to record the creator's address of kernel, system and RAM semaphores. If it is not used it is set to 0xffffffff.

**ChildWait BlockIDs** might be missed by the Dump Formatter and Kernel Debugger. Look out for **BlockIDs** of the form **0xffca????**. **ChildWait** is correctly reported from fix pack 29 for Warp V3 and base Warp V4.

Some **Sem32 BlockIDs** are missed by the Dump Formatter. Check **TCB\_SemInfo** for a 32-bit semaphore handle and **BlockIDs** of the form **0xfe?????**

If **BlockID** is a linear address owned by [ksem](#) then the semaphore is a **Kernel Semaphore**. However, not every **KSEM** is owned [ksem](#) owned memory. Under the ALLSTRICT kernel, a **KSEM** may be readily identified from the first 4 bytes, which have the signature "KSEM" Under the **RETAIL** and **HSTRICT** kernels, the **Blockid** is chosen to be the address (or handle) of the **KSEM**. Under the **ALLSTRICT** kernel, Event **KSEMs** use the **KSEM** handle+4 as the **BlockId**. Use [.D KSEM](#) command with the **BlockID** to format a **KSEM**.

In general a **BlockID** will be chosen to be meaningful to the programs using it. Often it is an address of a resource that needs to be serialised. Where no other information is given one should try:

	<code>.M <i>BlockID</i></code>	to try to establish an owner of the resource represented by the <b>BlockID</b>
	<code>LN <i>BlockID</i></code>	to try to establish a meaningful symbol associated with the <b>BlockID</b>
	Unwind the User's Stack	to try to establish the API or call the lead to the thread waiting on the <b>BlockID</b> (see <a href="#">.K command</a> )
<b>Addr</b>	<p>The address of the semaphore associated with this <b>BlockID</b></p> <p>See <b>Type</b> field discussion above for more precise information.</p>	
<b>Symbol</b>	<p>Either the symbolic address of the creator or of the associated semaphore.</p> <p>See <b>Type</b> field discussion above for more precise information.</p>	

## .PQ - Display Scheduler Queue Information



Display scheduler thread queue information for all (active) [threads](#).

### Syntax:

```
.PQ                                     ,
                                     #
                                     *
                                     slot
```

### Parameters:

*slot*

Display queue status for [thread slot](#) *slot*.

The following short-hand may be used for the slot number:

\*                      The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

`_TaskNumber`

#                      The debugger default thread slot. This defaults to the current slot unless overridden by the [.S command](#).

If no slot number is given then all thread slots are displayed in slot number order.

### Results & Notes:

The **.PQ command** locates each thread's [TCB](#) from the thread slot table, the linear address of which is given by global variable:

## **\_papTCBSlots**

or by traversing the process tree using **TCBpTCBNext** (**TCB +0x14**), **TCBpPTDA** (**TCB +0x08**) and **ptda\_pTCBHead** (**PTDA + 0x20**) fields.

Output from the **.PQ** command appears in tabular form as follows:

Slot	QSt	Pri	pTCB	PriNextQ	PriPrevQ	PriHigh	PriLow	PriNext	PriPrev
0001	blk	0100	ffe3c61c						
0002	blk	0200	7b9c8020						
0003	blk	0200	7b9c81d8						
0004	blk	081f	7b9c8390						
0005	blk	0800	7b9c8548						
0006	blk	0800	7b9c8700	7b9cb3b0	7b9c9830				
0008	blk	0500	7b9c8a70						
000a	blk	0800	7b9c8de0						
*000b	blk	0800	7b9c8f98	7b9ca960	7b9ca960				
000c#	blk	0800	7b9c9150	7b9cab18	7b9cab18				

Each of the fields has the following meaning:

### **slot**

The unique (hexadecimal) index in to the thread slot table of all threads.

This value may be flagged with:

- \*** to the left to signify the current (or last) dispatched thread.
- #** to the right to signify the Kernel Debugger default thread slot.

**Slot** may be found in the **TCBNumber** (**TCB + 0x2**) field of the **TCB**

### **QSt**

The thread's descending or current [scheduler](#) state taken from the **TCBQState** (**TCB +0x160**) field.

Except when a state transition is progress this is the same as the desired state of the thread (see **Sta** field of the [.P command](#).)

The following states are possible:

<i>Abrv</i>	<i>Qst</i>	<i>TCBQState Value</i>	<i>Description</i>
---	STATE_VOID	0	Uninitialized or Dead thread
rdy	STATE_READY	1	Thread ready to run
blk	STATE_BLOCKED	2	Blocked on a <a href="#">Block Id</a>
sus	STATE_SUSPENDED	3	*** Not in Use ***
crt	STATE_CRITSEC	4	Blocked by another CritSec thread (after attempting to run)
run	STATE_RUNNING	5	Thread currently running
tsd	STATE_TSD	7	Thread waiting for the TSD daemon to page in the TSD.
bst	STATE_READYBOOST	6	current state never set to this value - see note below.
dly	STATE_DELAYED	8	Delayed TKWakeup (Almost Ready)
frz	STATE_FROZEN	9	Frozen Thread via DosSuspendThread, DosCreateThread, DosExecPgm or DosSystemService
gsk	STATE_GETSTACK	10	TSD daemon is waiting for the page manager to page in a TSD

**Notes:**

**STATE\_READYBOOST** is a modified ready state and never becomes the current state, instead a priority boost is applied and the state becomes **STATE\_READY**

See **Sta** field description of the [.P Command](#) for related information on thread states.

**Pri**

Thread priority in **TCBPriorty** (**TCB** +0x168)

This is the current priority calculated by the scheduler based upon priority class (**TCBPriClass** (**TCB** +0x164)) priority class level (**TCBPriLevel** (**TCB** +0x165)) and priority boosts (**TCBPriClassMod** (**TCB** +0x166)). See **Pri** field description of the [.P command](#) for further information.

**pTCB**

Linear address of the [TCB](#) control block that represents the thread.

**PriNextQ**

The TCB address of the thread at the head of the next [priority queue](#).

**PriNextQ** is taken from the **TCBpTCBPriNextQ** (**TCB** + 0x170) double-word field.

If there are no other linked priority queues then **TCBpTCBPriNextQ** and **TCBpTCBPriPrevQ** point to this thread and **PriNextQ** and **PriPrevQ** are shown blank.

All TCBs not heading a priority queue have **TCBpTCBPriNextQ** and **TCBpTCBPriPrevQ** pointing to themselves.

**PriNext** and **PriPrev** is only of relevance to blocked and delayed threads.

**PriPrevQ**

The TCB address of the thread at the head of the previous [priority queue](#).

**PriPrevQ** is taken from the **TCBpTCBPriPrevQ** (**TCB** + 0x174) double-word field.

If there are no other linked priority queues then **TCBpTCBPriNextQ** and **TCBpTCBPriPrevQ** point to this thread and **PriNextQ** and **PriPrevQ** are shown blank.

All TCBs not heading a priority queue have **TCBpTCBPriNextQ** and **TCBpTCBPriPrevQ** pointing to themselves.

**PriNext** and **PriPrev** is only of relevance to blocked and delayed threads.

**PriHigh**

The TCB address of the next higher priority thread within this [priority queue](#).

**PriHigh** is taken from the **TCBpTCBPriHigher** (**TCB** + 0x178) double-word field.

If there are no higher priority threads on this priority queue then **TCBpTCBPriHigher** points to this thread and **PriHigh** is shown blank.

**PriLow**

The TCB address of the next lower priority thread within this [priority queue](#).

**PriLow** is taken from the **TCBpTCBPriLower** (**TCB** + 0x17c) double-word field.

If there are no lower priority threads on this priority queue then **TCBpTCBPriLower** points to this thread and **PriLow** is shown blank.

**PriNext**

The TCB address of the next thread of the same priority within this [priority queue](#).

**PriNext** is taken from the **TCBpTCBPriNext** (**TCB** + 0x180) double-word field.

If there are no other threads of the same priority on this priority queue then **TCBpTCBPriNext** and **TCBpTCBPriPrev** point to this thread and **PriNext** and **PriPrev** are shown blank.

**PriPrev**

The TCB address of the previous thread of the same priority within this [priority queue](#).

**Priprev** is taken from the **TCBpTCBPriPrev** (**TCB** + 0x184) double-word field.

If there are no other threads of the same priority on this priority queue then **TCBpTCBPriNext** and **TCBpTCBPriPrev** point to this thread and **PriNext** and **PriPrev** are shown blank.

-----

## Scheduler Priority Queues

Threads are linked in structures call **Priority Queues** or **PriQs**.

Priority queues are a double-linked list of thread priority groups. Each group is a double-linked list of threads of the same priority.

Six chain pointers are used for the links of a **PriQ**:

**TCBpTCBPriHigher** (**TCB** + 0x178)

**TCBpTCBPriLower** (**TCB** + 0x17c)

**TCBpTCBPriNext** (**TCB** + 0x180)

**TCBpTCBPriPrev** (**TCB** + 0x184)

By default these chain pointers are set to point to their own TCB.

**TCBpTCBPriHigher** and **TCBpTCBPriLower** link the heads of each priority group.

**TCBpTCBNext** and **TCBpTCBPrev** link the TCBs within each priority group.

A number of **PriQs** are defined. Each is anchored from a global symbol pointer:

<b>_ptcbPriQTSD</b>	Anchor for all threads in <b>tsd</b> state.
<b>_ptcbPriQRunner</b>	Anchor for all threads in <b>run</b> state. At most this contains one TCB.
<b>_ptcbPriQReady</b>	Anchor for all threads in <b>rdy</b> state.
<b>_ptcbPriQGetStack</b>	Anchor for all threads in <b>gsk</b> state.
<b>_ptcbPriQBadStack</b>	Anchor for all threads in <b>bad</b> state.
<b>ptda_pTCBPriQCritSec</b>	Anchor per-process for all threads within a process in <b>crt</b> state.

### Notes:

For the current process **ptda\_pTCBPriQCritSec** (**PTDA** +0x2e4) is a also a global symbol. Out of current context it can be located relative to the process' **PTDA** address.

The **TCB** address of the thread that has entered critical section is saved in **ptda\_pTCBCritSec** (**PTDA** +0x2e0)

Sleeping threads are queued on priority queues but in a manner to favour wake-up processing. The **Block-Id** is hashed to form an index into a table of **PriQ** anchors. The table is located at global symbol:

**\_aptcbSleep**

Each anchor points to a chain of **PriQs** of threads sleeping on the same **Block-Id**. The head TCB of each **PriQ** within a hashed chain is doubly linked from:

**TCBpTCBPriNextQ** (**TCB** + 0x170)

**TCBpTCBPriPrevQ (TCB + 0x174)**

Threads that happen to sleep on the same Block-Id as a multi-wake-up block-id are guaranteed not to be put in the same chain as the multi-wake-up threads.

When multi-wake-up threads wake their entire sleeping **PriQ** is moved to a chain of **PriQs** for threads in **dly** state. The delayed thread **PriQ** is anchored from global symbol:

**\_ptcbPriQDelayed**

Since **ptcbPriQDelayed** anchors a chain of **PriQs**, the head of each **PriQ** is doubly-linked using **TCBpTCBPriQNextQ** and **TCBpTCBPriQPrevQ**.

## .PU - Display Thread User Space Information



Display thread user space summary information for all (active) **threads**.

### Syntax:

**.PU**

#  
\*  
slot

### Parameters:

**slot**

Display user information for **thread slot slot**.

The following short-hand may be used for the slot number:

\* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

**\_TaskNumber**

# The debugger default thread slot. This defaults to the current slot unless overridden by the **.S command**.

If no slot number is given then all thread slots are displayed in slot number order.

### Results & Notes:

The **.PU command** locates each thread's **TCB** from the thread slot table, the linear address of which is given by global variable:

**\_papTCBSlots**

or by traversing the process tree using **TCBpTCBNext (TCB + 0x14)**, **TCBpPTDA (TCB + 0x08)** and **ptda\_pTCBHead (PTDA + 0x20)** fields.

Output from the **.PU** command appears in tabular form as follows:

Slot	Pid	Ord	pPTDA	Name	pstkframe	CS:EIP	SS:ESP	cbargs
0001	0001	0001	ffe3c7d4	*ager	ffe3bf54	1e30:00001794	0030:0000a402	0000
0002	0001	0002	ffe3c7d4	*tsd				
0003	0001	0003	ffe3c7d4	*ctxh				
0004	0001	0004	ffe3c7d4	*kdb				
0005	0001	0005	ffe3c7d4	*lazyw				
0006	0001	0006	ffe3c7d4	*asynchr				
0008	0002	0001	7b9e4020	pmsHELL	7b7d7f4c	d02f:0000272d	001f:0003f8b8	0008
*000a	0002	0002	7b9e4020	pmsHELL	7b7dbf44	d087:00003413	bfff:000007a6	0010
000b#	0002	0003	7b9e4020	pmsHELL	7b7ddf48	d087:0000351a	bfff:00000fc0	000c



Each of the fields has the following meaning:

**slot**

The unique (hexadecimal) index in to the thread slot table of all threads.

This value may be flagged with:

- \*** to the left to signify the current (or last) dispatched thread.
- #** to the right to signify the Kernel Debugger or Dump Formatter default thread slot.

**Slot** may be found in the **TCBNumber** (**TCB** + 0x2) field of the **TCB**

**Pid**

The [process id](#) this thread slot is assigned to.

**Ord**

The thread ordinal for this thread slot. This is the unique [thread id](#) assigned to the thread within the process to which it belongs.

**Ord** is taken from the **TCBOrdinal** (**TCB**+ 0x0) field of the

**pPTDA**

Linear address of the control block representing the process to which this thread belongs. The address is taken from **TCBpPTDA** (**TCB** +0x08).

**Name**

The name of the primary executable running in this process.

See **name** field description of the [.P command](#) for further information.

**pstkframe**

The address of the ring 0 stack frame that saved the user (ring 2 or ring 3) registers at the last transition to ring 0. For internal threads that have never run in ring 2 or ring 3 or for the currently running ring 3 thread this field will appear blank.

The address for the user stack frame is taken from **TCB\_pFrameBase** (**TCB** + 0x3c). See [.R command](#) for further information on displaying registers saved in the user stack frame.

**CS:EIP**

The user (ring 2 or ring 3) CS:EIP saved in the ring 0 user stack frame the last time the thread made a transition to ring 0. This field will appear blank if the thread is an internal ring 0 thread, currently running in ring 3 or the [TSD](#) for this thread is paged out. See [.I command](#) for information on paging in a TSD.

**SS:ESP**

The user (ring 2 or ring 3) SS:ESP saved in the ring 0 user stack frame the last time the thread made a transition to ring 0. This field will appear blank if the thread is an internal ring 0 thread, currently running in ring 3 or the [TSD](#) for this thread is paged out. See [.I command](#) for information on paging in a TSD.

**cbargs**

The user (ring 2 or ring 3) [cbargs](#) saved in the ring 0 user stack frame the last time the thread made a transition to ring 0. This field will appear blank if the thread is an internal ring 0 thread, currently running in ring 3 or the [TSD](#) for this thread is paged out. See [.I command](#) for information on paging in a TSD.

-----

## .R - Display User's Registers



Display the user registers for a given [thread slot](#). Set default addresses for [E command](#), [D command](#), [K command](#) and [U command](#).

Applicable to the Dump Formatter only, the default addressing mode is not set according to the **VM** flags of the **EFLAGS** register but is assumed always to be in protect mode. This has been corrected from fix pack 29 of Wapr 3.0 and base Warp 4.0.

**Syntax:**

.R

```
#
*
slot
```

### Parameters:

#### *slot*

Display user registers for [thread slot](#) *slot*. This option is valid **only** under the Kernel Debugger.

The following short-hand may be used for the slot number:

\* The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

`_TaskNumber`

# The debugger default thread slot. This defaults to the current slot unless overridden by the [.S command](#).

If no slot number is given then the debugger's default slot number is assumed.

### Results & Notes:

Registers are displayed and register mnemonics are assigned the values displayed for use in address expressions and operands of other Kernel Debugger and Dump Formatter commands.

The register information is obtained as follows:

Under the Kernel Debugger, if the displayed slot is the current system slot and the system is not in kernel mode (that is, **Indos**=1) then the hardware register values save by the debugger are displayed.

Otherwise the registers are extracted from the from the ring 0 stack frame base pointed to by **TCB\_pFrameBase** (**TCB** + 0x3c) for the thread slot in question.

The ring 0 stack frame base is created when the threads makes a transition from ring 2 or 3 to ring 0. This happens for a variety of reasons, such as issuing a call gate, trapping, pre-emption, interrupt, etc.. The format of the stack frame base depends on the reason for the ring 0 transition. **TCB\_pcriFrameType** (**TCB** + 0x38) points to the **CRI**, which contains a table of **RIPs**. Each **RIP** entry is associated with a specific hardware register. The **RIP** contains the offset and length of the associated register saved in the stack frame base. See [Client Register Information and Stack Frames](#) for details of the **CRI** and **RIP** formats.

#### **Note:**

If the thread has never run out of kernel mode, as is the case with some internal threads, then the **CRI** is never updated. The **.R** command is not able to format the user registers. For these threads the [R command](#) should be used, but only when the thread in question is the current system thread. Because the **R** command is an alias to the **.R** under the Dump Formatter, there is no way the display the current registers for an internal thread under the Dump Formatter. The only recourse is to display the **TSD** for the thread and attempt to unravel the stack manually.

If an invalid [thread slot number](#) is given the the Kernel Debugger issues the following message: prompted with the command prompt.

```
Invalid task number: nnnn
```

The format of the **.R** command output depends on whether the [RT command](#) has been used to toggle register display to full or short form and also whether the [Y 386ENV command](#) has been used to toggle register interpretation into 286 or 386 mode. Examples of the various forms follow:

```
##rt
##.r 2c
eax=f110099f ebx=00000001 ecx=0133fe4c edx=00000007 esi=0133ffec edi=00000000
eip=00000626 esp=0133fe20 ebp=0133fe88 iopl=2 -- -- -- nv up ei ng nz na pe nc
cs=d02f ss=099f ds=0053 es=0053 fs=150b gs=0000 cr2=1581928c cr3=001d0000
```

```
gdtr=7c3e5000 1fff idtr=ffe00df0 03ff tr=0010 ldtr=0028 cr0=pg et ts em mp --
dr0=00000000 --e1- dr1=00000000 --e1- dr2=00000000 --e1- dr3=00000000 --e1-
tr6=00000 v=0 d=00 u=00 w=00 c=w tr7=00000 ht=0 rep=0 dr6=-- -- -- dr7=-- --
002c|d02f:00000626 66ead77a021a5b00 jmp 005b:1a027ad7
```

```
##rt
##.r
eax=00000000 ebx=00000014 ecx=0000abd7 edx=0000abd7 esi=00080bff edi=00080007
eip=0000272d esp=0000a668 ebp=0008a670 iopl=2 -- -- -- nv up ei ng nz na pe nc
cs=d02f ss=0047 ds=abd7 es=d137 fs=150b gs=0000 cr2=1581928c cr3=001d0000
doscall11:CONFORM16:postDOSSEMWAIT:
d02f:0000272d c9 leave
```

```
##y 386env
##.r 2c
ax=099f bx=0001 cx=fe4c dx=0007 sp=fe20 bp=fe88 si=ffec di=0000
ip=0626 cs=d02f ds=0053 es=0053 ss=099f -- nv up ei ng nz na pe nc
002c|d02f:0626 66ead77a021a5b00 jmp 005b:1a027ad7
##
```

```
##rt
##.r 2c
ax=099f bx=0001 cx=fe4c dx=0007 sp=fe20 bp=fe88 si=ffec di=0000
ip=0626 cs=d02f ds=0053 es=0053 ss=099f -- nv up ei ng nz na pe nc
gdtr=3e5000 1fff idtr=e00df0 03ff tr=0010 ldtr=0028 iopl=2 msw=ts em mp
002c|d02f:0626 66ead77a021a5b00 jmp 005b:1a027ad7
##
```

Following the formatted register display, one line of disassembled code is displayed at the default disassembly address. [See the U command](#) for details on disassembling code.

Each of the fields has the following meaning:

### General Registers

These comprise the following registers:

ax, bc, cx, dx, sp, bp, si, di

eax, ebx, ecx, edx, esp, ebp, esi, edi

Each is displayed with its value in hexadecimal.

### Segment Registers

These comprise the following registers:

cs, ds, es, fs, gs, ss

Each is displayed with its selector value in hexadecimal.

### Instruction Pointers

These comprise the following registers:

ip & eip

Each is displayed with its value in hexadecimal.

### Flag registers

These comprise the following registers:

flags & eflags

These have their bit setting interpreted as follows:

Bit	Value	Flag	Description
17	1	VM	Virtual 8086 Mode (EFLAGS only)

16	0	RF	Resume Flag - Disable Debug Exceptions (EFLAGS only)
14	1	NT	Nested Task
11	1	OV	Overflow
11	0	NV	¬Overflow
10	1	DN	Direction Down
10	0	UP	Direction Up
9	1	EI	Enable Interrupts
9	0	DI	Disable Interrupts
7	1	NG	Negative Sign
7	0	PL	Plus Sign
6	1	ZR	Zero Result
6	0	NZ	Non-zero Result
4	1	AC	Auxiliary Carry
4	0	NA	¬Auxiliary Carry
2	1	PE	Parity Even
2	0	PO	Parity Odd
0	1	CY	Carry
0	0	NC	¬Carry

Bits 12 and 13 are the I/O Privilege Level bits. These are formatted as **iopl=level**.

Flags 14, 16 and 17 when reset are indicated by --.

### Memory Management Registers

<b>gdt</b> =xxxxxxxx yyy	Global Descriptor Table Register base address (xxxxxxxx) and limit (yyy)
<b>idt</b> =xxxxxxxx yyy	Interrupt Descriptor Table Register base address (xxxxxxxx) and limit (yyy)
<b>ldtr</b> =xxxx	Local Descriptor Table Register GDT selector (xxxx).
<b>tr</b> =xxxx	Task Register GDT selector (xxxx).

### Control Registers

<b>cr0</b> =	System control flags and Machine Status Word.
	These have their bit setting interpreted as follows:

Bit	Value	Flag	Description
31	1	PG	Paging Enabled
4	1	ET	Extension Type Flag - x87 support
3	1	TS	Task Switch Flag

2	1	EM	Emulation exception
1	1	MP	Math Present
0	1	PM	Protect Mode Enabled

Reset flag bit are shown with --.

**cr2=**

Page fault linear address.

**cr3=**

Page Directory Base Register (**PDBR**).

### *Debug Registers*

**dr0 to dr3**

These are formatted as follows:

```
dr0=11111111 glxnb
dr1=11111111 glxnb
dr2=11111111 glxnb
dr3=11111111 glxnb
```

where *11111111* is the breakpoint linear address and *glxnb* are **dr7** and **dr6** related flags.

The flags have th following interpretations:

***g***                **G** indicates a globally enabled breakpoint.

***/***                **L** indicates a locally enabled breakpoint.

***x***                **E** indicates an execute breakpoint

**R** indicates an read breakpoint

**W** indicates an write breakpoint

***n***                The number of bytes tested (1, 2 or 4)

***b***                **B** indicates a that a debug exception was generated that matched this breakpoint. This is the **B*n*** value of **dr6**.

- indicates a flag bit reset.

**dr6=**

The control bits 13-15 are interpreted as follows:

<i>Bit</i>	<i>Value</i>	<i>Flag</i>	<i>Description</i>
15	1	BT	Breakpoint triggered on task switch
14	1	BS	Breakpoint triggered on single step.
13	1	BD	Breakpoint on debug register access/update.

Flag bits not set are indicated by --

**dr7=**

The control bits 8 and 9 are interpreted as follows:

<i>Bit</i>	<i>Value</i>	<i>Flag</i>	<i>Description</i>
9	1	GE	Exact data matching enabled for global breakpoints
8	1	LE	Exact data matching matching for local breakpoints

Flag bits not set are indicated by --

### Test Registers

**tr6=***////* **v=***v* **d=***dd* **u=***uu* **w=***ww* **c=***c*

*////* is the linear page address.

*v* is **tr6** flag bit 11, the **valid bit**.

*dd* are **tr6** flag bits 10 and 9.

*uu* are **tr6** flag bits 8 and 7.

*ww* are **tr6** flag bits 6 and 5.

*c* is set as follows:

**r**                      **tr6** flag bit 0 set. **TLB** read command.

**w**                      **tr6** flag bit 1 reset. TLB write command.

**tr7=***ppppp* **ht=***h* **rep=***r*

*ppppp* is the **tr7** physical frame address.

*h* is flag bit 4 value. This is the **hit** or **PL** bit.

*r* are **tr7** flag bits 3 and 2. These are the **report** or **REP** bits.

The following **INTEL(R)** publications should be consulted for definitive information on processor registers:

**Intel486(TM) Microprocessor Family Programmer's Reference Manual**

**Pentium(TM) Processor User's Manual**

-----

## .REBOOT - Restart the System



Restart the system.

### Syntax:

.REBOOT

### Parameters:

*None*

## Results & Notes:

DosHlp service **DosHlpReboot** is called to restart the system. No system shutdown processing, whatsoever, is performed.

This command is not available to the Dump Formatter!

-----

# .S - Set or Display Default Thread Slot



Set or display the Kernel Debugger's and Dump Formatter's default [thread slot](#).

This command affects the default operation of the following:

D command  
E command  
U command.  
.I command  
.K command  
.P command  
.PB command  
.PQ command  
.PU command  
.R command

## Syntax:

```
.S          S          *          ~
              slot
```

## Parameters:

*slot*

Set the default [thread slot](#) to *slot*.

The following short-hand may be used for the slot number:

\*                      The current (last) thread the dispatcher gave control to. This value is taken from the word a global label:

TaskNumber

If no slot number is given **.S** displays the current thread slot number in message:

Current task number: *nnnn*

where *nnnn* is the thread slot number.

**S**

Set current ESP, EBP, SS, CS and EIP registers to those of the Dispatcher.

This option sets these registers as if the thread context had just been switched by the OS/2 Dispatcher. The [R command](#) will show the thread in kernel mode, about to be run.

No actual updating of register values takes place. Only default values are effected.

The new defaults are derived as follows:

**ESP**                      taken from **TSDKernelESP** (TSD + **Disp** value of [.P command](#) output.)

**EBP** taken from **TSDUserSSPad** (TSDKernelESP - 2)

**SS** selector 30 (**TASKAREA** segment).

**CS** Selector 170 (**DOSHIGH32CODE** segment).

**EIP** label **pgSwitchRet**.

This option is not available to the Dump Formatter.

**Note:**

The intent of this option is to simulate the correct value of the ring 0 stack selector for the default thread. This is only safe to use in commands that make explicit reference to the stack selector, for example:

```
.SS 21
R
DD SS:ESP
```

If an indirect reference is made to selector 30, for example by referring to a symbol from the **TASKAREA** segment then the adjustment to the default slot is not made. For example:

```
DW JFN_Table 114
```

will only display the **JFT** for the current thread slot. To display the **JFT** for another slot requires the following technique:

```
DW %ptda_address+JFN_Table-ptda_start 114
```

**Results & Notes:**

The **.S** command sets certain default values such that the view of the user's space in the new default slot is as if the thread **context** had switched. Linear and **LDT** selector based addresses will be accesses correctly by the Dump Formatter and Kernel Debugger. However, certain system data that are updated by a context switch are not changed and continue to display in the system's current thread context. These items include:

Task Register (TR)

GDT descriptor table entries for selectors 28, 30, 38 and 150b

Current **TSS** ring 0, and ring 2 stack selectors and pointers.

Global and System copy of the Current Local Information Segments.

The Thread Local Memory Area and Local Information Segment mapped by LDT descriptor **dfff**.

**Note:**

Descriptor **dfff** maps a global shared memory object, but it's data is copied from the incoming **PTDA** and **TCB** when a context switch occurs. This achieves the effect of thread local memory.

-----

## **.SYSDUMP - Force a System Dump and Restart the System**



Dump and Restart the system.

**Syntax:**

```
.SYSDUMP
```



**Parameters:**

*None*

**Results & Notes:**

The Kernel Debugger gives control to **RASRST** to force a system dump. The dump drive will be that specified in variable: **DumpDevice**, which defaults to **A**. This may be modified by using the **E command**.

The current CPU register values as displayed by the **R command** may be found from symbol **\_RegSA** and formatted using the **PMDf** REXX EXEC **DR**. See [Forcing a System Dump from the Kernel Debugger](#) for further information. **No system shutdown processing, whatsoever, is performed.**

This command is not available to the Dump Formatter!

-----

## .T - Dump the System Trace Buffer



Dumps the system trace buffer.

**Syntax:**

```
.T          count          MAJ=mm          ~
                                     MIN=nn
                                     ~
          S          filespec          ~
```

**Parameters:**

***count***

The number of trace entries to print, starting with the most recent. If not specified then the entire trace buffer will be dumped.

***MAJ=mm***

Specifies that only trace events with major code *mm* should be displayed.

See [System Trace Facility - Major Code Assignments](#) for a information on the deployment of trace major and minor codes in OS/2.

**Warning:** The Kernel Debugger may fail to process the **MAJ=** parameter correctly. Under some circumstances the debug kernel may hang. Use this option advisedly!

***MIN=nn***

Specifies that only trace events with minor code *nn* should be displayed.

This option required the specification of a major code using the **MAJ=** parameter.

See [System Trace Facility - Major Code Assignments](#) for a information on the deployment of trace major and minor codes in OS/2.

**Warning:** The Kernel Debugger may fail to process the **MAJ=** parameter correctly. Under some circumstances the debug kernel may hang. Use this option advisedly!

***S***

Specifies that the trace buffer should be saved to a file named in *filespec*.

**This option is only available to the Dump Formatter.**

The saved trace file may be subsequently formatted using the OS/2 [TRACFMT command](#).

### *filespec*

The file specification for the saved trace buffer.

The *filespec* may be fully qualified. The path defaults to the current directory.

### Results & Notes:

The trace is activated using the [OS/2 TRACE command](#).

If the trace is not active then the following message is generated:

```
Trace not on
```

The trace buffer is allocated in a single segment ([STDA](#)) whose selector may be located from global symbol **ras\_stda\_addr**. The STDA is a circular buffer whose entries are recorded in reverse order. The header gives the offsets to the first, last and current entries. The format of the trace buffer is described under [System Trace Data Area](#).

The major codes being traced are recorded in a bit string located at label **ras\_mec\_table**. Each active major code has its corresponding bit set.

The minor codes being traced are recorded in a bit string whose selector is located at label **ras\_min\_table**. The minor code table contains 32 byte entries, each corresponding to a major code. Each bit of each entry corresponds to a minor code within the major. If the bit is set then the minor code is traced.

When tracing by [Pid](#) is active then the **ptda\_rasflag** (PTDA +0x39a) is set to **0xff**.

The status of system tracing is recorded in status byte at label **ras\_systr\_flags**. The following flags are defined:

<i>name</i>	<i>bit mask</i>	<i>description</i>
RF_TRCAVAIL	0x80	System Trace Available
RF_TRCPAUSED	0x40	Trace paused
RF_TRCPID	0x20	Trace by PID
RF_TRCERRCOUNT	0x10	Tracing until error count
RF_TRCSUSPEND	0x08	Suspend due to error count
RF_TRCMINORCD	0x04	Tracing by Minor Code

Under the kernel debugger the system trace buffer cannot be saved directly. However by setting the **RF\_TRCPAUSED** bit in **ras\_systr\_flags** flag byte, the trace may be suspended and saved at a later time by using one of the system trace utilities (TRSPool, TRACEFMT or TRACEGET) or from a system dump. When setting **RF\_TRCPAUSED** be certain to OR in the flag bit.

.T command output appears as follows (see note at the end of this section for information on recent changes to the format of trace output):

```
MAJ=04 MIN=0089 PID=0006 CONTEXT=KERNEL:PROTECT
MAJ=06 MIN=008c PID=0000 CONTEXT=KERNEL:PROTECT
00 00
MAJ=06 MIN=000c PID=0000 CONTEXT=KERNEL:PROTECT TS=1336
08 00
MAJ=04 MIN=0009 PID=0006 CONTEXT=KERNEL:PROTECT
MAJ=04 MIN=0089 PID=0006 CONTEXT=KERNEL:PROTECT
```

Each of the fields is defined as follows:

#### **MAJ=**

The traced event major code.

#### **MIN=**

The traced event minor code.

#### **PID=**

The current [Pid](#) when the event occurred. See [.P command](#) for information on displaying active Pids.

**CONTEXT=***system:processor*

The system and processor [context](#) under which the event was traced.

*system* context may be:

<b>KERNEL</b>	If the trace record was created internally by a kernel routine.
<b>API</b>	If the trace record was created externally by use of the <b>DosDynamicTrace</b> or <b>DosSysTrace</b> APIs.  See <a href="#">Dynamic Trace Customiser</a> for information on creating dynamic trace records (via <b>DosDynamicTrace</b> ).  See <a href="#">DosSysTrace (Static Trace Event Recording)</a> for information on creating static trace records.

*processor* context may be:

<b>PROTECT</b>	If the trace record was created when the system was running in protect mode.
<b>REAL</b>	If the trace record was created when the system was running in real mode.

**TS=***hhss*

The system timestamp where *hh* is 100th seconds and *ss* is seconds.

The timestamp is taken from the [Global Information Segment \(GISEG+0xa\)](#). It is only recorded in the trace record if the time has changed since the previous timed stamped record was recorded.

**Note:** **TRACEFMT** treats this value as a word length fixed number of two decimal places.

*trace data*

Additional trace data.

A trace event may be accompanied with additional trace data, in which case it is dumped in hexadecimal and ASCII format on the following line.

Related information on the system trace facility may be found in:

- [System Trace Facility](#)
- [Dynamic Trace Customiser](#)
- [OS/2 Command reference - TRACE Command](#)
- [OS/2 Command reference - TRACEFMT Command](#)
- [OS/2 Command reference - TRACEBUF CONFIG.SYS statement](#)

### New Trace Format

From OS/2 2.11 fix pack 91 and OS/2 3.0 fix pack 8, the system trace has been enhanced to include more useful timestamp information. The Kernel Debugger and Dump Formatter were updated in fix packs 16 (OS/2 3.0) and 105 (OS/2 2.11) to take account of the new format.

### **Warning:**

The use of the **.T** command after the new trace format was implemented, but before the Kernel Debugger and Dump Formatter were updated, caused the Kernel Debugger and Dump Formatter to trap.

The following is an example of the new format:

```
#
Trace On at 0000,0000,0000,0000,0000,0000,0000
Trace Off at 0000,0000,0000,0000,0000,0000,0000
MAJ=03 MIN=0009 PID=0000 CONTEXT=KERNEL:PROTECT TS=3611,382e
00 00 00 00 bd 55 f5 ff 60 01 00 00 02 00 01 00 ....=Uu.`.....
MAJ=03 MIN=000f PID=0000 CONTEXT=KERNEL:PROTECT TS=3611,382e
00 00 cc cc f1 27 00 00 00 10 00 00 06 02 01 00 ..LLq'.....
c8 3c f2 ab H<r+
MAJ=03 MIN=0008 PID=0017 CONTEXT=KERNEL:PROTECT TS=3611,252d
00 00 00 00 93 86 e5 1b 5b 00 00 00 02 22 01 00 .....e.[...."..
MAJ=03 MIN=0008 PID=0017 CONTEXT=KERNEL:PROTECT TS=3611,222d
```

```

00 00 00 00 93 86 e5 1b 5b 00 00 00 02 22 01 00 .....e.[...."..
MAJ=03 MIN=0008 PID=0017 CONTEXT=KERNEL:PROTECT TS=3611,222d
00 00 00 00 93 86 e5 1b 5b 00 00 00 02 22 01 00 .....e.[...."..

```

The formatted trace is headed by a pair of time-stamps that give the time tracing was initiated and terminated. These are of the form:

YYYY,xxMM,xxDD,xxHH,xxmm,xxss,xxhh

where:

YYYY is years,

MM is Months

DD is Days,

HH is hours,

mm is minutes,

ss is seconds.

hh is 1/100th seconds,

xx ignore.

The time-stamp of each trace record is now shown as a pair of word values of the form:

TS=MMHH,hhss

where

MM is minutes,

HH hours,

hh 1/100s seconds and

ss seconds.

**Note:**

The byte reversal occurs because the time values are originally byte values which are displayed as words.

## System Trace Facility - User Guide

The OS/2 Trace facility is an important RAS mechanism within the OS/2 product. It allows specific events within the operating system, in system extensions and in applications to be recorded in a circular System Trace buffer. Software developers can create tracepoints that are used to monitor the execution of their software modules. A tracepoint is in essence a "window" that can be used to "peek" into a software module whenever that module reaches the state that corresponds to the tracepoint.

The OS/2 Trace facility includes three important utility programs:

### TRACE

The [OS/2 Trace control utility](#) is used to control (that is, enable and disable) the tracing of individual events.

### TRCUST

The [Dynamic Trace Customiser](#) is used to define both dynamic tracepoints for .DLL modules and trace formatting files for use by **TRACEFMT** utility.

### TRACEFMT

The [Trace Formatter](#) is used to format the contents of the sytem trace buffer for viewing. It may also be used for saving both formatted and unformatted trace data in a disk file and therefore may be used with the Dump Formatter

[.TS command](#) to format a system trace buffer that has been captured in a system dump.

The following topics are discussed in this chapter:

[Dynamic versus Static Trace](#)

[Guidelines for Defining Tracepoints](#)

---

## Dynamic Versus Static Trace

OS/2 supports two types of tracepoints:

- Static tracepoints
- Dynamic tracepoints

Both types of tracepoints can be used to monitor the execution of a software module. They differ in their style of execution.

Static tracepoints are, essentially, in-line function calls that are always present. The OS/2 user has the ability (through the use of the OS/2 TRACE utility) to indicate that a particular static tracepoint is "enabled". Until a static tracepoint is enabled, the actual tracepoint logic is "branched over" and not executed. Once "enabled" the static tracepoint logic is executed.

Dynamic tracepoints do not normally reside within the software modules to which they correspond. They are "patched in" when the OS/2 user uses the TRACE utility to "enable" a dynamic tracepoint.

This implies that a dynamic tracepoint does not burden a software module with any execution overhead until the tracepoint is "patched in" by the TRACE utility. There is a small continual performance overhead associated with a static tracepoint because the tracepoint must always check to see whether it is currently enabled. On the other hand, the dynamic tracepoint mechanism is costlier in operation than the static tracepoint operation because its "patching" mechanism is built upon the OS/2 breakpoint mechanism. Also, there are some software modules (for example, device drivers) that cannot use dynamic tracepoints. Most of the operating system tracepoints listed in the [Trace Reference](#) are implemented as dynamic tracepoints.

All Trace event records include a major trace code and a minor trace code which identify the event which is being recorded. When the user uses The TRACE utility to control a specific tracepoint, static tracepoints are identified by a combination of major and minor trace codes. Dynamic tracepoints are identified by a combination of software module name and minor trace code.

The following general categories of events are traced within OS/2:

1. External application program interfaces (APIs)
2. Internal interfaces
3. Other internal events

---

## Guidelines for Defining Tracepoints

The following subsections provide guidelines for programmers who are adding tracepoints to their software modules.

- [Assignment of Major and Minor Codes](#)
- [Trace Event Call Location](#)
- [Trace Event Parameters](#)

---

## Assignment of Major and Minor Codes

Component and subsystem developers who assign their own major and minor trace codes should follow the following conventions:

- 1. Minor codes should be assigned in the range 0001H-FFFFH. Minor code 0000H is reserved and should not be used.
- 2. An interface/event which requires both a pre-invocation trace and post-invocation trace of the event should share the same major code. The minor code assigned for the pre-invocation trace should be in the range 0001H-7FFFH. The minor code for the post-invocation trace of the event should be in the range 8001H-FFFFH and should be assigned by turning on the high order bit of the pre-invocation minor code. For example:

Event Name	Major Code	Minor Code
-----	----	----
KbdCharIn Pre-Invocation	0011H	0001H
KbdCharIn Post-Invocation	0011H	8001H

- 3. The Trace Definition Files for dynamic tracepoints should use the following TYPE definitions when defining tracepoints. Other TYPES or GROUPs may be defined as desired but these should be used as a minimum.
  - TYPE=(PRE,API) - pre-invocation of external interface
  - TYPE=(POST,API) - post-invocation of external interface
  - TYPE=(PRE,INT) - pre-invocation of internal interface
  - TYPE=(POST,INT) - post-invocation of internal interface

## Trace Event Call Location

Trace events, each with a unique minor code, should be placed at the following locations within a component or subsystem performing a service as a result of an external API call:

- 1. Pre-Invocation: At the beginning of any external interface before performing any actions or changing any data. This is considered pre-invocation of a service.
- 2. Post-Invocation: Before returning from performing any service through an external interface. This is considered post-invocation of a service, and is significant since:
  - a. Information may be returned to the caller which is of interest for problem determination
  - b. It causes a "service completion" event to be added to the trace buffer which may be paired with the pre-invocation trace event
- 3. Other: Events within a component should be considered as points of interest for tracing when they may be used for problem determination within the component or subsystem. These include:
  - a. Significant changes in the state of the machine.
  - b. Significant changes in the state of the system. System state changes are usually those which affect system data structures. These structures are those which are referenced by more than one component or which represent significant system resources.
  - c. Allocation or deallocation of a resource managed by a component performing a service

This detail is determined and assigned by each component or subsystem owner.

## Trace Event Parameters

In general, interface parameters should be passed with the trace call. Interfaces to performance sensitive code may require that only a few of the most important parameters, if any, be passed with the trace call.

1. Pre-Invocation:
  - a. Call by value parameters set by the caller should be passed with the trace call
  - b. Call by reference parameters to simple data (BYTE, WORD, DWORD) set by the caller should be passed and copied with the trace call
  - c. Call by reference parameters to character string data set by the caller should be passed and copied with the trace call
  - d. Call by reference parameters to complex data (structures other than character strings) should be handled by passing a subset of the most important data in the structure if the entire structure can not/need not be recorded
2. Post-Invocation:
  - a. A return code should always be returned from a post-invocation trace call. If the return code is not zero, then any additional parameters that are traced may not be valid.
  - b. Call by value parameters returned by the service should be passed with the trace call
  - c. Call by reference parameters to simple data (BYTE, WORD, DWORD) returned by the service should be passed and copied with the trace call
  - d. Call by reference parameters to character string data returned by the service should be passed and copied with the trace call
  - e. Call by reference parameters to complex data (structures other than character strings) should be handled by passing a subset of the most important data in the structure if the entire structure can not/need not be recorded
3. Other:
  - a. Significant data relating to the event
  - b. Address of a control block; Id or address of a resource

-----

## Dynamic Trace Customiser (TRCUST) - Reference

OS/2 provides a mechanism by which developers may dynamically apply tracepoints in their module at run time. This method eliminates all overhead of tracing when tracing is disabled. It also allows the developer to add tracepoints without modifying source code. This reduces the possibility that adding a tracepoint will induce errors into one's code. OS/2 needs a binary file, for each module being dynamically traced, which defines the tracepoints for the module.

### **Note:**

Information given here refers to the following versions of the system tracing tools except where explicitly noted:

TRCUST 3.06 or higher

TRACE 2.4 or higher

TRACEFMT 2.4 or higher

TRSPPOOL 4.2 or higher

TRACEGET with OS/2 Warp V3.0 FP35, OS/2 Warp V4.0 FP10 or OS/2 Warp E-Server or higher

[DTRACE](#) 4.3 or higher

[TRSPPOOL](#) 4.1 or higher

[TFFLST](#) 1.1 or higher

[TDFLST](#) 1.7 or higher

[DEBDEL](#) 1.0 or higher

[MAKETSF](#) 1.2 or higher

[MAPTSF](#) 1.7 or higher

**Note:**

There are certain restrictions on the use of dynamic trace which should be noted. These are:

1. Dynamic tracepoints may be applied to any non-VDM protect-mode module. However, in order to apply tracepoint to non-DLL modules a version of the trace command shipped with OS/2 Warp V3.0 fix pack 32, OS/2 Warp V4.0 fix pack 1 or OS/2 Warp E-Server is required.
2. Dynamic tracepoints may be applied to routines that run at interrupt time only after fix pack 35 for OS/2 Warp V3.0, fix pack 1 of OS/2 Warp V4.0 or OS/2 Warp E-Server.
3. A dynamic tracepoint cannot be applied to a module running under the Kernel Debugger that has a Breakpoint in place at the same location as the trace point.

The Trace Customizer (TRCUST) converts tracepoint definitions from a trace source file (TSF) into dynamic tracepoints for the trace definition file (TDF), and into formatting rules in the trace format file (TFF).

TRCUST provides a high-level access to Dynamic Trace, particularly suitable for use by developers. The associated [DTRACE](#) tool provides a very low-level interface to Dynamic Trace, which is suitable for attacking complex problems. Full details of the Dynamic Trace facility may be found in [The Dynamic Trace Facility And The DTRACE Tool](#).

**Definitions**

.TSF	An ASCII file created by a developer which defines all dynamic tracepoints for a given module. TRCUST allows just one major code to be associated with a TSF.
.TDF	A binary file, created by TRCUST, using the <b>.TSF</b> file as input. This file defines all tracepoints in the module in a manner acceptable to OS/2. This is used by the Trace Utility, TRACE.
.TFF	A binary file also created by TRCUST using the <b>.TSF</b> file. This file defines how all tracepoints will be formatted. This is used by the Trace Formatter, TRACEFMT.
major code	A byte value used to identify the module being traced. TRCUST allows at most only one major code per TSF.
minor code	A word value used to uniquely identify each tracepoint.
GROUP	A value used to identify this tracepoint with tracepoints of the same category. Examples are MEM for memory management and PM for Presentation Manager. For an example of uses of groups, see the online help for the <b>TRACE</b> command.
TYPE	A value used to associate a subset of dynamic trace events within a module. Examples are PRE for pre-invocation, POST for post-invocation and API for API calls within a module. For an explanation and examples of uses of types, see the online help for the <b>trace</b> command.

---

## Overview

[File Naming and Location](#)

[Invoking the Trace Customizer](#)

---

## File Naming and Location

The TDF file name is the same as the module to be traced, but has a file extension of "TDF". The TFF has a name of the form



"TRC00xx.TFF" where xx is the major code, for example, a module with major code 0xC2 will generate a TFF with the name "TRC00C2.TFF". This naming convention is used to allow TRACEFMT to dynamically generate the TFF name given only the major code.

TRCUST can be invoked to process a TSF or to combine several TFF files into a single TFF. For processing a TSF, TRCUST is given the name of a TSF, and optionally:

- the desired name of the resulting TDF
- the MAP file name
- the error message level

TRCUST will store the TSF tracepoint formatting specifications in the TFF file and if the tracepoint specified was not for a static tracepoint, the TSF tracepoint definition will be converted into the format required by the Trace Utility and stored in the TDF file. On errors, TRCUST will display appropriate messages, skip any tracepoint with errors in its definition, and continue processing the next tracepoint definition.

For combining TFF files that use the same major code, TRCUST is given the name of the file containing the TFF filenames to combine and the name of the file to contain the combined trace format statements.

TRCUST will issue an error message and abort processing under the following conditions:

- the TSF cannot be opened
- when combining TFF files, if any TFF input files cannot be read or if all TFF input files do not use the same major code
- when defining dynamic tracepoints, if the executable module to contain the tracepoints cannot be read
- the TDF, or TFF files cannot be written to
- an error in the header definition in the TSF
- a missing ending quote in the TSF

**Note:** TRCUST always returns 0 so that, when invoking it from a makefile, processing of the rest of the makefile can continue if TRCUST aborts.

Combine TFF files when several modules that use dynamic tracing use the same major code. The Trace Formatter can only use one TFF file per major code to get formatting information from. After the TSF file for each module is run through TRCUST to produce a TDF and TFF file, TRCUST can be invoked again, this time using the combine TFF files option and a file that only contains the paths to all the TFF files using the same major code. TRCUST will read all the TFF files. If all TFF files don't use the same major code, TRCUST will issue an error message and abort. TRCUST will read each trace format record from the TFF files and write them (in ascending order according to minor code) to the destination TFF file given.

---

## Invoking the Trace Customizer

The Trace Customizer is a protect mode only program and must therefore be run under OS/2.

TRCUST operates in two possible modes:

It may be invoked to combine TFF files,  
or to process a TSF.

---

## Invoking TRCUST to Combine TFF files.

The syntax for combining TFF files is as follows:

```
[d:][path]TRCUST [d:][path]tffsource /C=[d:][path]tffdest [/Wn]
```

where:

TRCUST

is the name of the Trace Customizer program. A drive and path may optionally be specified to explicitly define the location of the Trace Customizer program, otherwise the program is searched for in the current directory, followed by looking along the path defined by the PATH environment variable.

[d:][path]tffsource

is the name of a file containing fully qualified (including extensions) pathnames of TFF files to combine. Each TFF file must use the same major code and each filename in the **tffsource** file is separated by white space. This will combine all TFF files for the same major code into a single TFF file. If duplicate minor code format definitions are found, the first format definition for the minor code remains valid, the duplicates are discarded and a warning message is issued. If no path is provided the **tffsource** file is searched for in the current directory, followed by using the current value of DPATH.

[d:][path]tffdest

is the name of the trace format destination file to store the combined trace format definitions.

/Wn (optional)

is the level of error messages to be displayed, where **n** can be 0, 1, or 2. The possible message levels are shown below along with the messages that each displays:

0	fatal and severe messages
1	fatal, severe, and error messages
2	all (fatal, severe, error, and warning) messages

A message level of 2 is the default.

An example of a tffsource file for using the combine TFF files option of TRCUST is:

```
\TFF\PROG1\TRC00C2.TFF \TFF\PROG2\TRC00C2.TFF
\TFF\PROG3\TRC00C2.TFF \TFF\PROG4\TRC00C2.TFF
```

To invoke TRCUST to combine TFF files using the above file as input (assume filename is \TFF\PROG\TFF00C2) and output the combined format statements into file \TFF\PROG\TR\TRC00C2.TFF is:

```
TRCUST \TFF\PROG\TFF00C2 /C=\TFF\PROG\TRC00C2.TFF
```

---

## Invoking TRCUST to Process a TSF file.

The syntax for processing a TSF file is as follows:

```
[d:][path]TRCUST [d:][path]tsf [[d:][path]tdf] [/M=mapfile] [/Wn] [/D]
[/L=loadmod] [/NODE] [/NOLN] [/RMn] [/RSnnn] [/I] [/P]
[/PREINV] [/RAS]
```

where:

TRCUST

is the name of the Trace Customizer program. A drive and path may optionally be specified to explicitly define the location of the Trace Customizer program, otherwise the program is searched for in the current directory, followed by looking along the path defined by the PATH environment variable.

[d:][path]tsf

is the name of the trace source file. If no file extension is provided then an extension of TSF is assumed. If no path is provided the trace source file is searched for in the current directory, followed by using the current value of DPATH.

[d:][path]tdf (optional)

is the name of the trace definition file to store the dynamic tracepoint definitions. If not specified, the TSF filename is used with an extension of TDF. If no file extension is provided then an extension of TDF is assumed.

/D (optional)

allows duplicate minor codes to be used. This is useful where there is no need to distinguish different tracepoints which create records of the same format. For example, multiple return points from a subroutine.

/I (optional)

allows case-insensitive references to MAP file symbols to be used in TSF TRACE statements.

/L=loadmod (optional)

specifies the load module path and file name to be read by TRCUST. If not specified TRCUST uses the specification in the MODNAME statement of the TSF. If no path information is given in the MODNAME statement TRCUST assumes uses current directory and DPATH to locate the load module.

MODNAME is required by the TRACE command to allow it to load and determine the traced module's handle. /L is useful in cases where the load module is built using a name that differs from the installed name or where path information is required on the MODNAME statement for the TRACE command which differs from the directory used when the TDF is built. See also the **/P** switch for TRCUST.

/M=mapfile (optional)

defines *mapfile* as the MAP file for this module. The name may be qualified by a drive/directory, otherwise it will be searched for in the current directory followed by the path specified by the DPATH environment variable. If specified as an option, the MAP file must exist and the filename extension must be MAP or TRCUST will abort processing. The mapfile will only be used if a symbol is not found in the symbolic debug information stored in the executable module.

/NODE (optional)

forces TRCUST to ignore debugging information even if present. This is provided for cases where

- a) the user wishes to use MAP symbols in preference to debugging information.
- b) the level of debugging information is not supported by TRCUST and errors are produced if it is used.

**Note:**

Debugging information is not a public standard. Some compilers may appear to emulate supported styles of symbolic debugging information while not actually doing so.

TRCUST supports the following debugging information styles:

IBM C and C6 CodeView NB00 and NB02 styles.

IBM CSet/2 HLL version NB03 style.

IBM CSet/2++ and IBM VisualAge V1 to V3 HLL version NB04 styles.

IBM ALP assembler Debugging Information (HLL version NB04).

Any compiler or assembler that conforms to the HLL NB00 - NB04 debugging information styles.

/NOLN (optional)

forces TRCUST to ignore module line number records but honor any other debugging information present. This option is provided for cases where line number information does not conform to supported specifications and causes unpredictable results. Currently TRCUST is limited to supporting no more than 64K of line number information. For exceedingly large source files this limit could be exceeded.

**Note:**

/NODE implies /NOLN

/P (optional)

allows the path information specified with MODNAME to be retained in the TDF. The default is to strip path information from the module name. With no path information the TRACE command will rely on PATH, LIBPATH or the command line specification to the TRACE command to locate the traced module.

/PREINV (optional)

allows a tracepoint to be placed after the MOV (E)BP,(E)SP instruction on entry to a routine where local variable references will be valid. This is intended for use where local data references are made from the TSF, but the tracepoint is defined by reference to the routine's MAP file symbol name. In general it is better to use number references since /PREINV relies on the style of code generated by the compiler used. See [name](#) specification of the TP statement for information on difference between MAP file symbol references and symbolic debugging information references.

**Note:** PREINV does not search for the equivalent ENTER instruction.

/RAS (optional)

allows major code 0 trace definitions to be generated.

**Note:** Major code 0 is reserved by the system for system trace internal processing. The /RAS option should not be used except for the express purpose for which it is intended.

/RMn (optional)

specifies the mode by which the RETEP keyword of the TSF TRACE statement will operate. Eight modes of RETEP operation are defined as follows:

- |   |   |
|---|---|
| 0 | Disallow RETEP tracepoints.   |
| 1 | determine the RETEP tracepoint directly from CodeView symbol records only.                                  |
| 2 | use mode 1 then search for a LEAVE+RET or POP EBP instruction sequences near the end of the traced routine. |
|   | This is the default RETEP processing mode.  |
| 3 | use mode 2 then search for an isolated LEAVE instruction near the end of the traced routine.                |
| 4 | use mode 2 then search for an isolated RET instruction near the end of the traced routine.                  |
| 5 | use modes 2 and 3 combined.   |

**Note:** Version 2.0 of TRCUST only uses this search mode but is less reliable than mode 2 alone.

**Warning:**

Use of RETEP can run the risk of having a tracepoint not generated on an instruction boundary. The result of which would be unpredictable and probably cause the traced program to trap. This risk is greatly reduced by allowing the RETEP mode to default to /RM2. If the user chooses to widen the RETEP criteria then appropriate validation of the resulting tracepoints must be made.

- |   |   |
|---|---|
| 6 | use mode 4 then search for an isolated JMP instruction near the end of the traced routine.                            |
| 7 | use mode 6 but search for an isolated LEAVE instruction near the end of the traced routine, before searching for JMP. |

**Note:**

Successful location of return tracepoints can only be guaranteed with unoptimised IBM C and C6 CodeView information where the return point is given explicitly. More advanced compilers do not necessarily generate a single return point. They may even generate common epilog code for multiple routines hence the following limitations should be noted:

Only the last return from a routine is located.

Only mode 1 can accurately determine the last return, however this is only available to code-view version 0 modules.

Searching for instruction sequences may result in a tracepoint erroneously being placed within an instruction.

It may not be possible to use RETEP to define the return tracepoint for modules that use private (particularly optimised) calling conventions.

The default mode is 2. This reasonably safe. Higher modes are less safe.

Use mode 5 for compatible behaviour with earlier versions of TRCUST. However, note that earlier versions of TRCUST erroneously permit certain tracepoints which are not permitted using mode 5.

To avoid possible errors, JMP instructions with opcode 0xff are not selected in modes 6 and 7.

For an alternative method of specifying return tracepoints refer to the TP keyword of the TSF TRACE statement and how to use source line number references.

/RP (optional)

modifies RETEP processing to allow Pascal Return instructions (RET n) to be included in the search criteria for return tracepoint location. Use this with 16-bit code where you know that pascal returns are generated.

/RSnnn (optional)

specifies how far RETEP will search from the end of a routine to find the return instruction sequence. This defaults to 18 bytes however RETEP will not search before the start of the routine.

/Wn (optional)

is the level of error messages to be displayed, where n can be 0, 1, or 2. The possible message levels are shown below along with the messages that each displays:

0	fatal and severe messages
1	fatal, severe, and error messages
2	all (fatal, severe, error, and warning) messages

A message level of 2 is the default.

---

## Symbolic Debug Support

[Source Level Symbolic Support](#)

[MAP File Support](#)

[Building a Module](#)

---

## Source Level Symbolic Support

If the module has been compiled and linked with the debug options:

1. **/Ti** on the C/Set2 or VisualAge language compiler for C source files
2. **/CO** or **DE** on the link command

then the Trace Customizer can look into the module to extract symbolic or debugging information. In this case addresses may be specified symbolically. Having generated trace definitions there is no further use for symbolic information. Thus debugging information may be safely removed by using the [DEBDEL](#) utility.

### Notes:

Not all source files must be C language when using symbolic support. Assembler and other languages may be used as long as they generate Microsoft 16-bit CodeView or IBM HLL version 3 and 4 symbolic debugging information.

Tracepoints may be defined by filename and line number references or by symbolic names.

Tracepoints may reference local or global variables.

If symbolic names are referenced then the following points should be observed:

Some compilers will modify external or public symbolic names used in the program. If this is done then the modified form must be used when referenced in the TSF. The modification rules are compiler dependent however three common cases are cited below.

Names must be used case sensitively.

With IBM C/2 and Microsoft C 6.0, external routine names have an underscore prefix to their names when declared using the **cdecl** convention and are capitalised when declared using the **Pascal** convention. Symbols for routine names treated this way refer to the first instruction of the routine. When debugging information is present, the unaltered name is also usable and refers to the first instruction following the prologue code, that is creation of the entry stack frame. For example both **\_main** and **main** may be used. The former refers to the true entry point and the latter after the prologue code. The advantage of using the latter form (**main**) is that symbolic references may be made to the parameters passed; in this example **argc** **argv** and **envp**.

C runtime library routine names are usually prefixed with an underscore.

With IBM CSET/2 and IBM VisualAge C symbolic names defined by the user are *not* prefixed with an underscore. However C runtime library routine names *are* usually prefixed with an underscore. Symbols for routine names defined by the user refer to the initial instruction of the routine when compiled without symbolic debugging information and to the first instruction following the prologue code when debugging information is present. In the latter case, symbolic references may be made to parameters passed to the routine. In the former case this is not possible except by making **EBP** relative references and using the **/PREINV** command line switch - see [MAP File Support](#) for more information in using symbolic references without the presence of symbolic debugging information.

With IBM VisualAge C++ routine names are subject to name *mangling* where class-hierarchical information is encoded into the externalised name. The mangled form of the name must be used in TSF references.

In all cases symbolic name lengths are restricted to a maximum length of 255 characters.

-----

## MAP File Support

The Trace Customizer can also use the symbolic information in the MAP file produced by the linker. All public symbols will be listed with their offsets in the module being traced. This is not as complete a support as offered by the debug compile option for C language source files, but it does allow entry points, public labels and global data to be referenced symbolically within the TSF. Note that the use of a MAP file is NOT language dependent.

### Notes:

When using a MAP file, if the symbolic name is a C language entry point, it will be case sensitive (unless the **/I** command line switch has been specified).

Some compilers will modify external or public symbolic names used in the program when they appear in the MAP file. If this is done then the modified form must be used when referenced in the TSF. The modification rules are compiler dependent however three common cases follow:

With IBM C/2 and Microsoft C 6.0, external routine names have an underscore prefix to their names when declared using the **cdecl** convention and are capitalised when declared using the **Pascal** convention. Symbols for routine names treated this way refer to the first instruction of the routine.

With IBM CSET/2 and IBM VisualAge C symbolic names defined by the user are *not* prefixed with an underscore. MAP file symbols for routine names defined by the user refer to the initial instruction of the routine, regardless of the presence of debugging information, unless the **/PREINV** command line switch is specified. When **/PREINV** is used TRCUST will search for the first instruction following the prologue code, that is, after the entry stack frame has been created.

With IBM VisualAge C++ routine names are subject to name *mangling* where class-hierarchical information is encoded into the externalised name. The mangled form of the name must be used in TSF references.

In all cases symbolic name lengths are restricted to a maximum length of 255 characters.

C runtime library routine names are usually prefixed with an underscore.

-----

# Building a Module

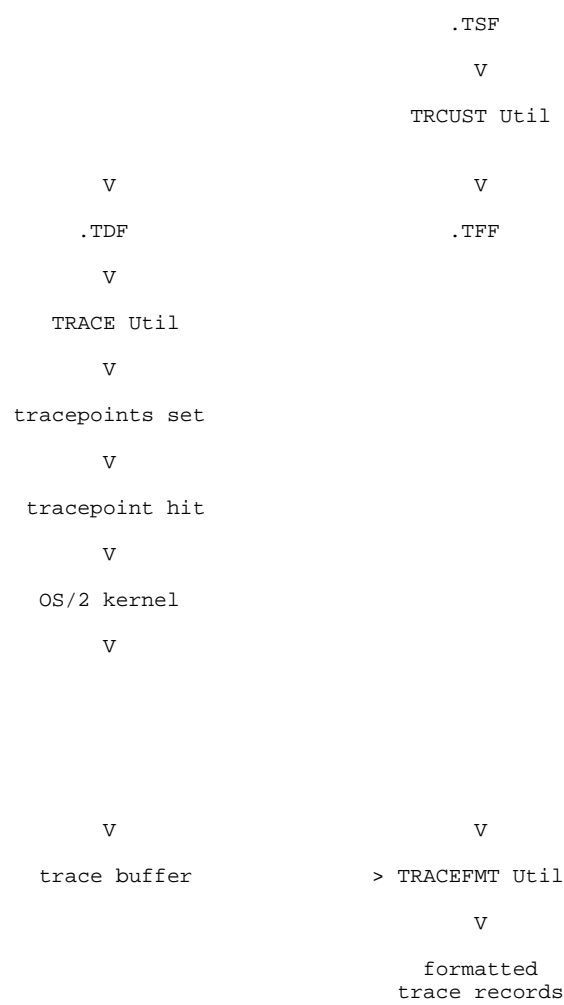
To trace only public procedures, you only need your MAP file that was generated by linking your module.

To trace local variables in C language routines, compile the C programs with the debug option and assemble the ASM routines with public symbols. Link all the OBJs together with the debug option (/CO) and run TRCUST on the executable module. You can now strip the debug information from the executable file by either relinking the OBJs without the debug option or by using the [DEBDEL](#) tool to delete the debug information from the executable module file.

-----

# TDF and TFF File Usage

The TDF, and TFF files produced by TRCUST are used in the following way:



## How TRCUST fits into the system

To trace a module do the following:

1. Define the tracepoints and data to be traced in the TSF.

2. Invoke the Trace Customizer using the TSF as input.

This produces two files, a TDF and a TFF.

3. Put the TDF file in the same directory the module to trace resides, put the TFF file in a directory accessible by TRACEFMT. It is suggested that all TFF files reside in the same subdirectory, an example directory would be \OS2\SYSTEM\TRACE, which is the default directory searched by TRACEFMT. However a private directory may be specified by using the */t=* command line option or the **Set TFF Path** menu pulldown of TRACEFMT.

4. Invoke the OS/2 TRACE command using either the name of the TDF or the module name instead of the major code value, which is done only with static trace.

**Note:**

The TRACE command will assume that a simple unqualified name refers to a TDF and will search, \OS2\SYSTEM\TRACE, the current directory and DPATH for the TDF. If it is still unable to locate the TDF then the name is assumed to be a DLL and LIBPATH is searched for the TDF.

If a qualified name is used then TRACE will assume this to be the traced module and will attempt to locate the TDF in the module's directory, by searching the current directory then LIBPATH for DLLs or PATH for non-DLLs.

This activates the tracepoints, causing the trace data to be saved in the system trace buffer.

5. The OS/2 TRACE command can be used to turn tracing off at any time.

6. The system trace buffer may be captured for formatting at a later time using either of the TRACEGET or TRSPOOL utilities. Or alternatively TRACEFMT may be used to capture and format the trace buffer immediately.

TRACEFMT uses the major code to determine the TFF file and uses the formatting string corresponding to the minor code value to format the data in the RAS trace buffer and output it to the screen, file or printer.

Tracepoints may selectively be turned on or off at any time, by reference to individual minor codes or a group or group+type combination.

Active tracepoints may be queried using either:

```
TRACE /Q
```

or

```
DTRACE QUERY /X /A
```

The latter example gives detailed information on tracepoint status and location within each module.

-----

## Symbols and Abbreviations Used in the Document

[...] denotes optional items.

[... | ... | ...] denotes a list of optional items, zero or more of which may be chosen.

{... | ... | ...} denotes a list of items of which ONE must be chosen.

item... denotes that *item* is repeated zero or more times.

statement,..... denotes this example is incomplete.

nnn is a number in the range 0-255 inclusive.

nnnnn is a number in the range 0-65535 inclusive.

All numbers and values can be entered in decimal form or in C hexadecimal form (0x....).

-----



# Trace Source File

This section details the statements that can appear within a trace source file.

Examples are given of TRACE statements.

## TSF Format

The layout of a trace source file is:

```
Header
Type List Definition
(optional)
Group List Definition
(optional)

Tracepoint Definitions
```

Layout of a trace source file

### Note:

- Comments may be freely inserted anywhere in the trace source file. A comment is identified by a ; or by using C syntax comments anywhere in the file. A C comment has start and end delimiters, namely /\* and \*/. C type comments may span lines, and may be nested.
- Below are sample TSF files. See [Sample Trace Source Files](#) for more examples.

```
; Sample trace source file depicting dynamic tracing for OS calls compiled
; with 32-bit addressing

MODNAME = doscall11.dll
MAJOR   = 100 /* this is decimal, would be 0x64 if specified hex */
MAXDATALENGTH = 200 /* max bytes logged per tracepoint is 200 */

TYPELIST NAME=PRE,ID=1,
        NAME=SYS,ID=0x40,
        NAME=API,ID=128, /* decimal, if hex would be 0x80 */
        NAME=POST,ID=0x8000

GROUPLIST NAME=MEM,ID=2,
        NAME=FS,ID=0x5,
        NAME=MOU,ID=13,
        NAME=DOS,ID=0x2B /* would be 43 if decimal */

TRACE MINOR=0x0001,
      TP=.DosOpen, /* Pre-invocation tracing on DosOpen */
      TYPE=(PRE,API),
      GROUP=DOS,
      DESC="(OS) DosOpen Pre-Invocation",
      FMT="Major = %X Minor = %Y",
```

```

        FMT="                EAX = %D",
        FMT="                FileName = %P%S",
        REGS=(EAX),
        ASCIIZ32=(.FileName,DIRECT,128)

TRACE    MINOR=0x7001,          /* Puts tracept on code at line 28 */
        /* of file dosopen1.c. Debug */
        TP=@dosopen1.c,28,    /* info is needed to use this. */
        TYPE=(API),
        GROUP=DOS,
        DESC="(OS) CheckParm After Createhandle",
        FMT="                New handle = %P%W",
        MEM32=(.handle,DIRECT,2)

TRACE    MINOR=0x8001,          /* Post-invocation tracing at */
        TP=.DosOpenC,RETEP,    /* procedure DosOpenC return point. */
        TYPE=(API,POST),      /* Debug info is needed to use */
        GROUP=DOS,            /* this type of tracepoint. */
        DESC="(OS) DosOpenC Post-Invocation",
        FMT="                Return Code = %P%W",
        FMT="                Variable Rec= %P%U",
        MEM32=(.retcode,DIRECT,2),
        /* The following will log a variable length structure. The */
        /* second field in the structure is the length of the */
        /* record(position var_struct+2). This LEN parameter must */
        /* immediately precede the memory specification defining */
        /* the variable length record. */
        LEN=(var_struct+2,DIRECT),
        MEM32=(.var_struct,DIRECT,LEN)

```

-----

## TSF Header

This defines common information for the module to be traced. The format is:

```

[ MODNAME = [d:][path]Name ]
[ MAJOR   = nnnnn ]
[ MAXDATALENGTH = nnnn ]
[ TDFID = nn ]

```

where:

MODNAME=[d:][path]Name where:

d:

is the drive containing the module. If not specified the current drive is used.

path

is the path to the module. If not specified the current path is used.

Name

is the name of the executable module to be traced. If an extension is not specified and the **Name** is not **OS2KRNL**, an extension of DLL is appended to **Name**.

Path information may be specified but is normally discarded before storing module name information in the TDF unless the **/P** switch is specified on the command line. TRCUST uses the path information to locate and read the load module while processing the TSF. If either the name or path differs from the module's installed name or path then the **/L=** command line option may be use to direct TRCUST on while module file name to read.

**Note:** MODNAME is required only for dynamic tracepoint definitions with TRCUST version 2.22 and later.

MAJOR=nnnnn (optional)

defines the major trace ID allocated to this module. It may be in the range 1 to 65535 decimal or specified 0x1 to 0xffff hex. The default value is 1. The major trace ID is part of the data placed in the trace buffer when a tracepoint is executed.

Only one major code is permitted per module.

**Notes:**

Prior to fix pack 35 for OS/2 Warp V3.0 and for OS/2 Warp V4.0 only major codes 0x1 - 0xff may be used. After fix pack 35 for OS/2 Warp V3.0 the major code range of 0x1 - 0xffff permissible. TRCUST version 2.2 or later is required if the extended range of major codes is to be used.

Also note that the default major code (1) is reserved by OS2 for dekkko tracepoints. Ranges available to the user are: 245-255 and 0x8000-0xffff.

Major code 0 is permissible but only with the **/RAS** switch specified on the command line. However this is reserved for specific use by system use and should not normally be used.

MAXDATALENGTH=nnnn (optional)

defines the maximum amount of data that a single tracepoint call will insert into the trace buffer.

The length may be in the range 20 to 4099 decimal or specified 0x14 to 0x1003 hex. The default value is 512.

**Note:**

Prior to fix pack 35 for OS/2 Warp V3.0 and fix pack 10 for OS/2 Warp V4.0 MAXDATALENGTH defaults to 128 and may only specify a value in the range of 20 to 512.

TDFID=nn (optional)

defines an identifier in the range 0 to 65565 for use with the GETVARS and QUERY functions of the [DTRACE command](#). The default TDFID is 0 and is normally allowed to default.

---

## Typelist Definition

This defines the optional typelist event IDs. For more description and examples of event types see the online help for the **trace** command.

The intent of type definitions is to assign one or more categories to a tracepoint so that they may be selected as a subset of one or more groups by the TRACE command. For example, tracepoints at routine entry points might be assigned a type of **PRE**, while those defined at return points might be assigned a type of **POST**. Each tracepoint may be associated with 0 or more types.

The format is:

```
TYPELIST NAME=TypeName, ID=TypeValue,
        [NAME=TypeName, ID=TypeValue, ]...
```

where:

NAME=Typename

defines a 1-8 byte character string used to reference the TypeValue in the tracepoint definitions. All TypeNames and GroupNames within a TSF must be unique.

ID=TypeValue

defines a bit value of the form 2\*\*y where y in range 0 to 15, permitting a maximum of 16 types to be defined in a single TSF. This can be decimal or specified 0xnnnn for hex.

An example TYPELIST definition follows:

```
TYPELIST NAME=PRE, ID=1,
        NAME=SYS, ID=0x40,
        NAME=API, ID=128,
        NAME=POST, ID=0x8000, .....
```

---

## Grouplist Definition

This defines the optional grouplist IDs. For more description and examples of groups see the online help for the **trace** command.

The intent of group definitions is to assign subsets of tracepoints to a meaningful category so that they may be selected by group name using the TRACE command. For example, tracepoints associated with file system APIs might be assigned to the **FS** group. Each tracepoint may be associated with at most one group. Note the difference between group and type association in this aspect.

The format is:

```
GROUPLIST NAME=GroupName, ID=GroupValue,
          [NAME=GroupName, ID=GroupValue, ]...
```

where:

NAME=GroupName

defines a 1-8 byte character string used to reference the GroupValue in the tracepoint definitions. There are a maximum of 48 GroupNames allowed in a TSF file. All TypeNames and GroupNames within a TSF must be unique.

ID=GroupValue

defines a word value in the range 1 to 65535 decimal or 0x1 to 0xFFFF hex.

An example GROUPLIST definition follows:

```
GROUPLIST NAME=MEM, ID=2,
          NAME=FS, ID=0x5,
          NAME=MOU, ID=13, .....
```

---

## Tracepoint Definitions

The tracepoint address and the data to be traced are specified by the **TRACE** statement. There are a maximum of 65535 tracepoints permitted in a trace source file.

The format of the TRACE statement is:

```
TRACE      [MINOR=minorcode,]
           TP={@STATIC, |@filename, linenum, |.name[ {+|-}offs][, RETEP[=retopt[+retopt...]]]},
           [OPCODE=0xnn,]
           [TYPE=(typename[, typename...]),]
           [GROUP=groupnam,]
           [DESC="Tracepoint description",]
           [FMT="Formatting string",]...
           [LEN=(length_spec, flag),]
           [DATA_STMT, ]...
```

The **TRACE** keyword delimits a tracepoint definition statement. The definition is considered complete when the next **TRACE** keyword is encountered or the end of file is reached. There is one **TRACE** statement for each tracepoint.

**LEN** is used to log variable length records. A *DATA\_STMT* must immediately follow the **LEN** statement. **LEN** will give the location of a one word field containing the number of bytes to log for the following *DATA\_STMT*.

---

# MINOR Keyword

The MINOR parameter is an optional keyword parameter. If it is specified in the first tracepoint definition, it must be specified in every tracepoint definition. If it is not specified in the first tracepoint definition, it cannot be specified in any of the subsequent tracepoint definitions. It should be coded as:

```
MINOR=nnnnn ,
```

where:

nnnnn

is a decimal number from 1 to 65535 or a hex number from 0x1 to 0xFFFF. This represents the minor code for the tracepoint, which should normally be unique within the major code specified for this module. When duplicates are encountered, the original trace definition is saved and duplicates are discarded with an error message.

Duplicate minor codes within major code are permitted when the **/D** command line switch is specified. However there are precautions to note:

Any formatting information specified in trace definitions with duplicate minor codes is discarded. Only the original is retained in the generated TFF.

Use of duplicate minor codes should be limited tracepoints where a common formatting template may be used. For example, multiple exit points from a routine.

If minor codes are not specified at all in the TSF then TRCUST will generate them sequentially for each trace definition encountered starting with 1. However, if any definition has an explicit minor code specified then all definitions must have explicit minor codes specified.

-----

# TP Keyword

The TP parameter is a required keyword parameter. If TP is specified more than once for a single tracepoint definition, the tracepoint is discarded. TP has three mutually exclusive definitions which can be coded as:

```
TP=@STATIC ,
```

where:

STATIC

defines this tracepoint entry to be used only for creating a trace format statement for the TFF file. No tracepoint definition is created for the TDF, and the only other TRACE parameters that will be used are **DESC**, **MINOR** and **FMT**. This is used to create trace formatting information for static tracepoints (or dynamic tracepoints generated directly by the [DTRACE command](#)). If the TSF contains only @STATIC directives, no TDF files are created and MODNAME is then not required.

```
TP=@filename,linenum,
```

where:

filename

is an ASCII string specifying the name (including extension) of a source filename used in creating the module. The source filename is stored in the debug information contained in the executable module, so debug information must exist to use this parameter. The

filename is not case sensitive.

**Note:**

path information may be specified with **filename** when also specified at compilation time.

linenum

is a decimal number specifying the line number in the given source file name to place the tracepoint.

**Note:** Debug information must exist to use this option. The statement at the given source linenum may have been rearranged during compiler optimization, so the developer must use this with caution. If the linenum is not found in the debug information, the tracepoint is applied at the next linenum defined in the debug information and a warning message is issued to the user.

An example to apply a tracepoint to line 35 of file stubfile.c is:

```
TRACE    MINOR=0x700A,                /* puts tracepoint on code at line */
         TP=@stubfile.c,35,.....    /* 35 of source file stubfile.c */
```

Use of line number reference affords great flexibility in tracepoint location, however the user will need to modify the TSF file every time referenced source files are updated. To facilitate this the [MAKETSF](#) utility may be used to generate line number references automatically from TSF statements embedded as comments in C or Assembler source.

```
TP=.name[ {+|-}offs][,RETEP[=retopt[+retopt...]]],
```

where:

name

is a public label or an entry point name of a procedure to be traced. The "." preceding **name** is required. **Name** must be found in the debug information in the module or **name** must be a public symbol as found in the MAP file. If debug information is used, the address of this tracepoint will be immediately following the prologue of the procedure. If MAP information is used, this address points to the opcode at the given label unless the **/PREINV** command line switch is specified. In this case TRCUST attempts to locate the instruction following the

```
MOV EBP,ESP
```

or

```
MOV BP,SP
```

instructions as with done when debug information is used. Use of **/PREINV** allows parameter and local variable references to be made relative to **EBP** in non-optimised code.

See [Source Level Symbolic Support](#) and [MAP File Support](#) for information on using symbolic references.

offs (optional)

is a decimal (specified as nnnnnnnn) or hex (specified as 0xxxxxxxxx) offset from the entry point address.

RETEP (optional)

specifies that the tracepoint will be inserted at the **return** address corresponding to this entry point.

Default criteria for specifying the return point are specified by the **/RM RETEP** mode command line option. TRCUST will search back from the end of the routine for the instruction sequence matching the RETEP mode. The length of the search is governed by the **/RS** command line option. The default RETEP mode may be overridden per tracepoint by coding one or more optional **retopt** keywords, separated by a + sign with RETEP. The following keywords are allowed:

CV use CodeView information.

LRET search for a LEAVE+RET or POP EBP + RET instruction sequences near the end of the routine.

RET search for RET instruction sequences near the end of the routine.

JMP search for JMP instruction sequences near the end of the routine.

**Note:** to reduce the hazard of generating a tracepoint on a non-instruction boundary, JMP instructions with 0xFF opcodes are not selected.

LEAVE search for a LEAVE instruction near the end of the routine.

**Note:**

If more than one option is specified the RETEP uses the following order of precedence: CV, LRET, RET, LEAVE, JMP.

When the RETEP is used, the **name** must be a valid entry point to a procedure.

The RETEP option depends upon the manner in which a C compiler generates its code. Therefore this option may not work with some of the newer compilers or with code optimisation.

**Warning:**

Use of RETEP can run the risk of having a tracepoint not generated on an instruction boundary. The result of which would be unpredictable and probably cause the traced program to trap. This risk is greatly reduced by allowing the RETEP mode to default to **/RM2**. If the user chooses to widen the RETEP criteria then appropriate validation of the resulting tracepoints must be made.

**Note:** For ASM functions to accomplish tracing, a label must be made public to have a tracepoint applied. Therefore, to accomplish "POST" tracing, a label must be made public at the return statement.

Partial examples of Pre/Post tracing of DosOpen follows:

```
TRACE    MINOR=0x0001,
          TP=.DosOpen,.....          /* Pre-invocation tracing */

TRACE    MINOR=0x8001,
          TP=.DosOpen,RETEP,.....     /* Post-invocation tracing */
```

**Note:** It is not possible to set dynamic tracepoints on the following machine instructions:

0x9C	PUSHF
0xCC	INT 3
0xCD	INT n
0xCE	INTO
0x62	BOUND
0x69	IMUL
0x6B	IMUL
0xF6	DIV, IDIV, MUL, IMUL
0xF7	DIV, IDIV, MUL, IMUL

TRCUST gives an error for these opcodes and the tracepoint is rejected.

**Note:** No more than one tracepoint may be applied to a given address.

---

## OPCODE Keyword

The OPCODE parameter is an optional keyword parameter.

```
OPCODE=0xnn,
```

where:

nn

is the expected one byte hex opcode to be found at the tracepoint address and TRCUST verifies the value with that in the module. The opcode of the instruction being traced must be the same as this value or an error message is issued and the tracepoint is rejected. This may be used to verify the opcode expected at the address specified by the **TP** parameter. This may be useful when using **TP = @filename,linenum** to ensure the requested instruction is traced.

---

## TYPE Keyword

The TYPE parameter is an optional keyword parameter that defines the event types of this tracepoint. For more description and examples of event types see the online help for the **trace** command. Also see the [TYPELIST](#) statement for information on how types are used.

```
TYPE=(typename[ ,typename... ]),
```

where:

typename

is an ASCII string specifying the type of this tracepoint. The typename symbol must have been previously defined by the **TYPELIST** statement. If an invalid typename is given, the tracepoint will be discarded and a message issued.

The final type value is obtained by logically combining each type name value using the OR operator. If **TYPE** is omitted, the trace statement will have a *typevalue* of 0.

---

## GROUP Keyword

The GROUP parameter is an optional keyword parameter that defines the group this tracepoint belongs to. For more description and examples of groups see the online help for the **trace** command. Also see the [GROUPLIST](#) statement for information on how types are used.

```
GROUP=groupnam,
```

where:

groupnam

is an ASCII string specifying which group this tracepoint belongs. The groupname symbol must have been previously defined by the **GROUPLIST** statement. If an invalid groupname is given, the tracepoint will be discarded and a message issued.

If **GROUP** is omitted, the trace statement will have a *groupvalue* of 0.

---

## DESC Keyword

The DESC parameter is used to produce a description for the tracepoint that is output as the first line of formatted data. It should include the entry point name of the procedure being traced and whether this is an entry or return point. The descriptive string is enclosed in double quotes as for a C language string. The DESC parameter is required if any FMT specifications are present.

The recommended formats for such strings are as follows:

```
DESC="name Pre-Invocation",
```

```
DESC="name Post-Invocation",
```

where:

name



is the system component (in parentheses) followed by the entry point name of the procedure.

#### Pre-Invocation

identifies this tracepoint as an entry point, that is, before the function has been executed.

#### Post-Invocation

identifies this tracepoint as a return point from the function.

The words *Pre-Invocation* and *Post-Invocation* are not mandatory, merely recommendations to be compatible with the base OS/2 tracepoints, when formatted. If a tracepoint is inserted in the middle of a procedure it will be appropriate to use different wording. The Trace Customizer does not check the wording.

An example of pre-invocation and post-invocation tracepoints follow:

```
TRACE  MINOR=0x0001,
        TP=.DosOpen,
        DESC="(OS) DosOpen      Pre-Invocation",.....

TRACE  MINOR=0x8001,
        TP=.DosOpen,RETEP,
        DESC="(OS) DosOpen      Post-Invocation",.....
```

-----

## FMT Keyword

The optional FMT parameter is used to produce the formatting string for the trace data. The developer should use these to control formatting the output produced by the Trace Formatter. Each **FMT** keyword causes CR/LF to be appended to the format string. The formatting string is similar to a C library printf string specification. It consists of ASCII characters and formatting controls enclosed in double quotes as for a C language string. Each formatting primitive describes the format of the data in the trace buffer at the formatting position and must match the data stored in the trace buffer by the data statements described later. See [Formatting Trace Data](#) for a description of how the data is stored in the trace buffer and subsequently formatted.

The formatting controls are as follows:

**%Innn** Ignore nnn number of bytes in the trace buffer.

This tells the Trace Formatter to skip over the next nnn bytes in the current trace record. This could be used, for example, to skip over unimportant data, traced as a block, and only output the data of interest.

When using this control, nnn represents an ASCII decimal number and must be followed by a space.

```
statement: FMT = "ignore ten bytes %I10 here",
           FMT = "          and two more %I2 here",

generates: ignore ten bytes here
           and two more here
```

**%P** Process the data prefix bytes associated with the trace data.

This tells the Trace Formatter that the next bytes in the trace record are the prefix or header bytes for data logged by the dynamic tracing mechanism. This is required to precede any format control describing data logged from memory. Do not use this before data that was logged from a register and never use with static tracepoints.

%P may be used in combination with any of the following formatting controls: %A, %B, %C, %D, %F, %Q, %S, %U, %W.

%P%x may be specified as %Px.

See [Formatting Trace Data](#) for a description of how the data is stored in the trace buffer and the use of this control.

```
statements: FMT="memory byte = %P%B",
```

```
generates: memory byte = C2
```

**%B** Output a byte of data in hexadecimal.

```
statement: FMT = "memory byte = %P%B"
generates: memory byte = 01
```

**%C** Output an ASCII character

```
statement: FMT = "memory bytes = %C%C%C"
generates: memory byte = ABC
```

**%C** formats a byte of data as a single ASCII character, with no separating spaces. Bytes outside the range 0x20 - 0x7f are formatted as periods.

**Note:**

The **%C** formatting control is only available with TRCUST 2.26 or later and TRACEFMT version 2.2 or later.

**%W** Output a word of data.

```
statement: FMT = "register word = %W"
generates: register word = 0001

statement: FMT = "memory word = %P%W"
generates: memory word = 0001
```

**%D** Output a double word of data.

```
statement: FMT = "double word EAX = %D"
generates: double word EAX = 0000 4B2C

statement: FMT = "double memory word = %P%D"
generates: double memory word = 0000 4B2C
```

**%F** Output a Flat (0:32 bit) address.

```
statement: FMT = "flat address EAX = %F"
generates: flat address EAX = 00004B2C
```

**%Q** Output a quad word of data.

```
statement: FMT = "quad word from regs EAX and EBX = %Q"
generates: quad word from regs EAX and EBX = 00004B2C 00000001
```

**%A**      Output a segmented (16:16 bit) address.

```
statement: FMT = "segmented address in SS:SP = %A"
```

```
generates: segmented address in SS:SP = 00B7:0001
```

```
statement: FMT = "segmented address in memory = %P%A"
```

```
generates: segmented address in memory = 00B7:0001
```

**%R**      Repeat the following format control for the rest of the memory that was logged.

The action of %R differs between dynamic and static trace:

For dynamic trace records, %R will process the 3-byte memory prefix to determine the length of data to which repeat formatting applies. %R is used in place of the %P control.

For static trace records, %R applies itself to all the remaining data in the trace buffer. No 3-byte memory prefix is examined or processed.

%R may be used only with the following formatting controls: %A, %B, %C, %D, %F, %Q, %W, in particular it has no meaning with %S and %U.

```
statement: FMT = "log a variable number of words from memory = %R%W"
```

```
generates: log a variable number of words from memory = 0001 0004
```

**%S**      Output an ASCIIZ string.

The prefix formatting control should always precede this for dynamic tracepoints because the data was logged from memory.

**Note:** If the tracepoint is static, then **%P** should not be used because the string is terminated with a null byte.

```
statement: FMT = "string = %P%S"
```

```
generates: string = c:\os2\os2.ini
```

**%U**      Format the remainder of the trace record as a sequence of bytes in hexadecimal or dump format depending upon the View pull-down selection from TRACEFMT.

This will output the remaining of the traced data, including any prefix bytes.

```
statement: FMT = "garbage = %U"
```

```
generates: garbage = [00 00 00 03 c2 c1 c4 ff 04 00 09 c0 18]
```

With fix pack 35 for OS/2 Warp V3.0, fix pack 10 of OS/2 Warp V4.0 and OS/2 Warp E-Server the TRACEFMT utility permits unformatted data to be displayed in dump format, where both the hexadecimal and ASCII representations of the data are displayed together with offsets. ASCII characters outside the range 0x20-0x7f are displayed as '.' for example:

```
Data [+0000  04 00 5A 00 00 0C 00 2B-DF F6 FF FF FF 00 00 01  ..Z....+..... ]
Data [+0010  00 01 C0                                     ... ]
```

**%X**      Output the major event code.

```
statement: FMT = "major code = %X"
```

```
generates: major code = 00C2
```

%Y      Output the minor event code.

```
statement: FMT = "minor code = %Y"
```

```
generates: minor code = 0081
```

**Note to CMVC Users:** To avoid conflicts with source file control information, all formatting specifications can be in upper or lower case. Also, prefix format specifications may be combined with data format specifications. For example, the following create the same format controls in the TFF:

```
FMT = "%P%W here"
FMT = "%p%w here"
FMT = " %P %W here"
```

---

## LEN Keyword

The LEN parameter is an optional keyword parameter that defines the length of the variable length record that will follow in the next MEM or MEM32 statement.

```
LEN=(length_spec,flag),
```

where:

length\_spec

is an address specification that points to the one word length field of the next memory specification. This format can be **symbolic\_name+nnnnnnnn** where **symbolic\_name** is a symbolic memory location and **nnnnnnnn** is the offset from that symbolic address. The **length\_spec** can also be [Flat Register Form](#). or [Segment Register Form](#).

flag

is a mandatory parameter that identifies the level of indirection to be used on the length\_spec. It is one of:

```
D[IRECT]
I[NDIRECT][*[{+|-}iiiiiii]]...
```

DIRECT implies that the length\_spec specifies a memory location that contains the length of the variable length record. INDIRECT means that the length\_spec contains an address and is dereferenced to obtain the memory location. The optional asterisks denote the level of indirection, one for each level. The indirect offsets **iiiiiii** are added to or subtracted from the value found at the given level of indirection.

The following are example LEN statements followed by the memory statement whose length they describe.

```
TRACE MINOR=.....,
/* Symbol vrecord is a record whose first field is a one */
/* word value that is the total length of the entire      */
/* variable length record.                                */
LEN=(vrecord,DIRECT),
MEM=(.vrecord,DIRECT,LEN),

/* Symbol vrec_ptr is a pointer to a variable length record */
```

```

/* and vend_ptr is a pointer to the end of the same record. */
/* The second field (10 bytes from end of record) is total */
/* length of the variable length record. */
LEN=(vend_ptr,INDIRECT*-10),
MEM=(.vrec_ptr,INDIRECT,LEN),

/* Symbol vrec_ptr is a pointer to a variable length record.*/
/* The second field (2 bytes from beginning of record) is */
/* total length of the variable length record. */
LEN=(vrec_ptr,INDIRECT*+2),
MEM=(.vrec_ptr,INDIRECT,LEN),

/* Symbol ind_ptr is a pointer to a structure. The third */
/* field in the structure (6 bytes from beginning) is a */
/* pointer to a variable record. The fourth field in the */
/* variable length record (8 bytes from beginning) is the */
/* total length of this variable length record. */
LEN=(ind_ptr,INDIRECT*+6*+8),
MEM=(.ind_ptr,INDIRECT*+6*,LEN),

/* If DS:DI contains the address of ind_ptr, to perform */
/* the above logging, the statements would be: */
LEN=(RDS+DI,INDIRECT*+6*+8),
MEM=(RDS+DI,INDIRECT*+6*,LEN)

```

-----

## DATA\_STMT

There are three types of data that may be traced as part of the optional DATA\_STMT section of the TRACE statement.

Registers  
Memory  
ASCIIZ strings

More than one keyword is permitted in a tracepoint definition. The order of the statements defines the order in which the data is inserted into the trace buffer.

The combined amount of data to be traced for a single tracepoint cannot exceed MAXDATALENGTH. If TRCUST determines that the maximum data size might be exceeded, a warning message is issued but the tracepoint definition will remain valid.

The keywords for tracing the three types of data are **REGS**, **MEM32**, **MEM**, **ASCIIZ32**, and **ASCIIZ**.

The **REGS** keyword identifies which registers are to be recorded in the trace buffer.

The **MEM32** keyword is used to record sections of memory in the trace buffer. Access to this memory location is through 32-bit flat addresses from functions compiled using 32-bit addressing. Several MEM32 parameters may be coded at any one tracepoint if several different memory areas are to be traced.

The **MEM** keyword is also used to record sections of memory in the trace buffer, but access to this memory is through a segment:offset pair. This is used for functions compiled using 16-bit addressing with segment registers. Several MEM parameters may be coded at any one tracepoint if several different memory areas are to be traced.

The **ASCIIZ32** keyword is used to record an ASCIIZ string in the trace buffer. This is a special form of the MEM32 keyword and there may be more than one ASCIIZ32 parameter coded for a single tracepoint.

The **ASCIIZ** keyword is used to record an ASCIIZ string in the trace buffer. This is a special form of the MEM keyword and there may be more than one ASCIIZ parameter coded for a single tracepoint.

-----

## REGS Keyword

This is coded as:

```
REGS=(register[,register]...),
```

where:

register

is one of the following to support OS/2 versions 1.1 and 1.2:

```
CS,DS,SS,ES,AX,BX,CX,DX,SP,BP,SI,DI,IP,FLAGS
```

with the addition of the following to support OS/2 version 2.0:

```
EAX,EBX,ECX,EDX,ESP,EBP,ESI,EDI,EFLAGS,EIP,FS,GS
```

or the symbolic name of a C language variable declared with the register storage-class specifier as:

**.symbolic\_name**

The same register may appear multiple times in the register list. It will be traced as many times as it appears. Extended registers (E) are 32 bits and logged as two words. All other registers are 16 bits and logged as one word.

**Note:** To log a C language variable declared with the register storage class, debug information must exist and the variable name is case sensitive. When formatting the data logged from a register variable, remember that there are no memory prefix bytes put into the log buffer.

Example of the REGS statement follows:

```
/* Given the following declaration in a C language source file: */
register int ret_code;

/* To log registers AX, CX and the register variable ret_code: */
TRACE MINOR=.....
REGS=(AX,CX,.ret_code),
FMT="AX=%W CX=%W ret_code=%W"
```

-----

## MEM32 Keyword

This is used to log memory in a function compiled using 32-bit flat addressing and is coded as:

```
MEM32=(address_spec,flag,{length|LEN}),
```

where:

address\_spec

is a flat memory address specification as described in [Address Specification](#).

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

```
D[IRECT]
I[NDIRECT][*[{+|-}iiiiiii]]...
IS
```

DIRECT implies that the address specifies a memory location to be saved in the trace buffer.

INDIRECT means that the address contains a flat address and is dereferenced to obtain the memory location. The optional asterisks denote the level of indirection, one for each level. The indirect offsets **iiiiiii** are added to or subtracted from the value found at the given level of indirection.

IS (Indirect Segmented) means that the address contains a segmented address that is dereferenced to obtain the memory location.

length

is the number of bytes at the memory location to be saved in the trace buffer. If *length* is too big, a warning message will be given, and *length* will be set to MAXDATALENGTH. If *length* is 0 an error message will be given, and this tracepoint will be ignored.

LEN

specifies that this is a variable length record to log and the length was specified by the preceding **LEN** statement. If there was no preceding **LEN** statement, this tracepoint is rejected. Either length or LEN must be specified, but not both.

Example of the MEM32 statement follows:

```
TRACE MINOR=....
/* To log retcode enter the following: */
MEM32=(.retcode,DIRECT,2),

/* s_ptr is a pointer to a structure, log it for 4 bytes. */
MEM32=(.s_ptr,INDIRECT,4),

/* Field 6 bytes into structure pointed at by s_ptr is a      */
/* pointer to a structure, log 8 bytes past begin of struct.*/
MEM32=(.s_ptr,INDIRECT*+6*+8,10), /* logs ten bytes */

/* s_ptr points to a variable length record, second field */
/* is the record length (offset 4 from record beginning).*/
LEN=(s_ptr,INDIRECT*+4),
MEM32=(.s_ptr,INDIRECT,LEN)

/* s_end points to the end of same variable length record,*/
/* second field is the record length (offset -6 from      */
/* record beginning).                                     */
LEN=(s_end,INDIRECT*-6),
MEM32=(.s_ptr,INDIRECT,LEN)
```

-----

## MEM Keyword

This is used to log memory in a function compiled using 16-bit segment:offset addressing and is coded as:

```
MEM=(address_spec,flag,{length|LEN}),
```

where:

address\_spec

is a segmented memory address specification as described in [Address Specification](#).

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

```
D[IRECT]
I[NDIRECT][*[{+|-}iiiiiii]]...
IF
```

DIRECT implies that the address specifies a memory location to be saved in the trace buffer.

INDIRECT means that the address contains a segmented address and is dereferenced to obtain the memory location. The optional asterisks denote the level of indirection, one for each level. The indirect offsets **iiiiiii** are added to or subtracted from the value found at the given level of indirection.

IF (Indirect Flat) means that the address contains a flat address that is dereferenced to obtain the memory location.

Only far pointers may be dereferenced when using segmented addressing.

length

is the number of bytes at the memory location to be saved in the trace buffer. If *length* is too big, a warning message will be given, and *length* will be set to MAXDATALENGTH. If *length* is **0** an error message will be given, and this tracepoint will be ignored.

LEN

specifies that this is a variable length record to log and the length was specified by the preceding **LEN** statement. If there was no preceding **LEN** statement, this tracepoint is rejected. Either length or LEN must be specified, but not both.

---

## ASCIIZ32 Keyword

This is used to log a string in a function compiled using 32-bit flat addressing and is coded as:

```
ASCIIZ32=(address_spec,flag,maxlength),
```

where:

address\_spec

is a 0:32 bit flat memory address specification as described in [Address Specification](#).

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

```
D[IRECT]
I[NDIRECT][*[{+|-}iiiiiii]]...
IS
```

DIRECT implies that the address points to a memory location, the contents of which are to be saved in the trace buffer.

INDIRECT means that the address points to a flat address pointer which is dereferenced to obtain the target location to save in the trace buffer. The optional asterisks denote the level of indirection, one for each level. The indirect offsets **iiiiiii** are added to or subtracted from the value found at the given level of indirection.

IS (Indirect Segmented) means that the address points to a segmented address pointer which is dereferenced to obtain the target location to save in the trace buffer.

maxlength

is the maximum length of the string that will be saved in the trace buffer. It should be no greater than MAXDATALENGTH. The actual length to be traced will depend on where the zero terminating byte is found.

If *maxlength* is **0** an error message will be given, and this tracepoint will be ignored.

**Note:** When using dynamic tracing, the OS/2 kernel *does not* place the terminating null byte into the trace buffer; therefore the prefix byte must be used by the Trace Formatter to obtain the length of the string.

---

## ASCIIZ Keyword

This is used to log a string in a function compiled using 16-bit segment:offset addressing and is coded as:

```
ASCIIZ=(address_spec,flag,maxlength),
```



where:

address\_spec

is a segmented memory address specification as described in [Address Specification](#).

flag

is a mandatory parameter that identifies the level of indirection to be used on the address. It is one of:

D[IRECT]  
I[NDIRECT][\*[{+|-}iiiiiii]]...  
IF

DIRECT implies that the address points to a memory location, the contents of which are to be saved in the trace buffer.

INDIRECT means that the address points to a far pointer which is a segmented address that is dereferenced to obtain the target location to save in the trace buffer. The optional asterisks denote the level of indirection, one for each level. The indirect offsets **iiiiiii** are added to or subtracted from the value found at the given level of indirection.

IF (Indirect Flat) means that the address points to a far pointer which is a flat address that is dereferenced to obtain the target location to save in the trace buffer. Only far pointers may be dereferenced using segmented addresses.

maxlength

is the maximum length of the string that will be saved in the trace buffer. It should be no greater than MAXDATALENGTH. The actual length to be traced will depend on where the zero terminating byte is found.

If *maxlength* is **0** an error message will be given, and this tracepoint will be ignored.

**Note:** When using dynamic tracing, the OS/2 kernel *does not* place the terminating null byte into the trace buffer; therefore the prefix byte must be used by the Trace Formatter to obtain the length of the string.

---

## Address Specification

The syntax for specifying a memory address given here applies to the MEM32, MEM, ASCII32 and ASCII keywords above.

An address is specified in one of the following forms:

1. Symbolic name form (can be used for MEM32, MEM, ASCII32, and ASCII).
2. Flat register form (can be used only for MEM32 and ASCII32).
3. Segment register form (can be used only for MEM and ASCII).

---

## Symbolic Name Form

This is coded as:

.name[{+|-}nnnnnnnn]...[{+|-}(iiiiiii)],

where:

name

is a symbolic name of a memory location. The "." is required before the **name**. The debug information in the module is checked for

the **name** and if not found and a MAP was given, the MAP is checked. An error message is output by the Trace Customizer if the symbol is not found and the trace definition is ignored.

The **name** is normally case sensitive and may be subject to modification by the compiler.

See [Source Level Symbolic Support](#) and [MAP File Support](#) for information on using symbolic references.

If the **/I** command line switch is specified then MAP file symbolic references may be case-insensitively.

**name** is limited to a maximum length of 255 characters. If a symbol exceeds this length then TRCUST will only use the first 255 characters. A warning message is issued when name truncation occurs.

nnnnnnnn (optional)

is a displacement from the symbolic address. If hex the syntax is 0xnnnnnnnn.

iiiiiii (optional)

is a displacement from the indirect address. If hex the syntax is 0xiiiiiii. This specifies a displacement from the final address when using INDIRECT, IF (Indirect Flat) or IS (Indirect Segmented) addressing.

-----

## Flat Register Form

This is coded as:

```
Fbreg[ {+|-}ireg ] ... [ {+|-}nnnnnnnn ] ... [ {+|-}(iiiiiii) ],
```

where:

breg

is a flat model (0:32 bit) base register and is one of:

EAX,EBX,ECX,EDX,ESP,EBP,ESI,EDI,EIP

ireg (optional)

is an extended data, base or index register. More than one ireg may be used to define a displacement from the flat register value to the memory location. It may be one of:

EAX,EBX,ECX,EDX,EBP,ESI,EDI,EIP

nnnnnnnn (optional)

is an optional fixed displacement to be added to the address calculated in the registers. If hex the format is 0xnnnnnnnn.

iiiiiii (optional)

is a displacement from the indirect address. If hex the syntax is 0xiiiiiii. This specifies a displacement from the final address when using INDIRECT or IS (Indirect Segmented) addressing.

This form of address is calculated at run time.

-----

## Segment Register Form

This is coded as:

`R sreg[ {+|-} dreg ] . . . [ {+|-} nnnnn ] . . . [ {+|-} (iiii) ] ,`

where:

sreg

is a segment register and is one of:

CS,DS,SS,ES,FS,GS

**Note:** The prefix **R** implies that the adjoined segment register is to be used with an initial offset of zero. Thus **RDS** refers to data addressed by **DS:0**

dreg (optional)

is a data, base or index register. More than one dreg may be used to define a displacement from the segment register value to the memory location. It is one of:

BP,SP,SI,DI,AX,BX,CX,DX,IP

nnnnn (optional)

is an optional fixed displacement to be added to the address calculated in the registers. If hex the syntax is 0xnnnn.

iiii (optional)

is a displacement from the indirect address. If hex the syntax is 0xiiii. This specifies a displacement from the final address when using INDIRECT or IF (Indirect Flat) addressing.

This form of address is calculated at run time.

-----

## Formatting Trace Data

This section gives a brief description of the formatting process as an aid to generating correct formatting strings.

Each trace record stored in the RAS buffer consists of a header followed by a number of variable length trace data records. The header identifies the major and minor code, time stamp, process ID, etc., and the total length of the trace data for that trace record.

Each **MEM32**, **MEM**, **ASCIIZ32**, or **ASCIIZ** data statement, coded in the trace source file for a tracepoint, produces an associated data record to be stored in the trace buffer. The data records consist of a 3-byte prefix followed by the trace data. This prefix consists of a status byte followed by the length of the data for that statement. The status byte indicates whether valid data has been traced.

Dynamic trace can only trace data that is resident in memory at the time that the tracepoint is executed. Data may not be able to be traced for two reasons: it resides in a page that is currently paged out or the address specified is invalid. This latter case usually occurs due to tracing indirectly via invalid pointer variables. In either of these two cases dynamic trace sets the status byte accordingly and stores the pointer in the place of the wanted data. No more data is attempted to be traced for this invocation of the tracepoint, but tracing will resume the next time this tracepoint is encountered.

Since the position of these prefix bytes, within a trace record, is dependent on the data being traced and the number of **MEM32**, **MEM**, **ASCIIZ32**, or **ASCIIZ** statements, the Trace Formatter must be told when to expect the prefix in the trace record. This is the purpose of the **%P** and **%R** formatting controls. One of these must be coded in the formatting string at every place a data record is expected, (noting the restrictions on the use of **%R** specified in the description of the **FMT=** keyword.

The following values are used in the status byte of the prefix:

0	Valid data to log
1	ASCIIZ string to log
-1	Invalid Selector
-2	Selector not Present
-3	Page not Present

If the prefix indicates an error TRACEFMT will interpret the error and format the failing address.

**Note:** With ASCIIZ and ASCIIZ32 commands, the prefix is used by the TRACEFMT utility to obtain the length of the string since the string is not null terminated.

---

## Sample Trace Source Files

This section gives four sample TSF files. The first is for a module written in a mix of C and MASM and compiled with 16:16 segmented addressing. The second was compiled with 0:32 flat addressing. The third module consists of routines, some which were compiled using 16-bit segmented addressing and some that were compiled using 32-bit flat addressing. The fourth is for monitoring function references in a module.

---

## TSF Using 16-bit Segmented Addressing

```
; Trace source file for the xxx dynalink.  Compiled with 16-bit offsets.

MODNAME=\c\src\xxx.dll
MAJOR=0xC5
MAXDATALEN=200
; We will want to trace up to 200 bytes in any one trace call.

TYPELIST NAME=API,ID=08,
          NAME=SYS,ID=04,
          NAME=PRE,ID=02,
          NAME=POST,ID=64

GROUPLIST NAME=MEM,ID=1,
          NAME=FS,ID=3

/* The following tracepoint does not need debug info,
   only a MAP file is necessary with label xxalloc
   public in it.  The program must be compiled in 16-bit
   mode because segmented addressing is used (ASCIIZ
   instead of ASCIIZ32).
   This logs the word registers AX and BX and the string
   pointed at by DS:DI for a max of 20 bytes. */

TRACE    MINOR=25, TP=.xxalloc,
          OPCODE=0x8B, /* the opcode is optional */
          TYPE=(API,PRE),
          GROUP=MEM,
          DESC="(OS) xxalloc Pre-Invocation",
          FMT = "                AX = %W ",
          FMT = "                upper BX = %B",
          FMT = "                lower BX = %B",
          FMT = "                param = %P%S",
          REGS=(AX,BX),
          ASCIIZ=(RDS+DI,DIRECT,20)

/* This defines a tracepoint at Foo label.  The ten words
   to log are found indirectly through SS:SP.  Note that
   each word needs a format control but since only one
   memory access was done, one prefix control is needed. */

TRACE    MINOR=0xB0, TP=.Foo,
          TYPE=(SYS),
          GROUP=FS,
          DESC="(OS) Foo Pre-Invocation",
          FMT="                First Five words = %P%W%W%W%W%W",
          FMT="                Three words ignored %I6",
          FMT="                Last Two Words = %W%W",
          MEM=(RSS+SP,INDIRECT,20)
```

```

/* This defines a tracepoint at Goo label. DS:DI points
   to a structure whose second field is a pointer to an
   ASCIIIZ string. The offset from the first field in the
   structure is 4 bytes. Max string size to log is 40 bytes. */

TRACE MINOR=0xB1, TP=.Goo,
      TYPE=(SYS),
      GROUP=FS,
      DESC="(OS) Goo Pre-Invocation",
      FMT="          Second field in struct points to %P%S",
      ASCIIIZ=(RDS+DI+4,INDIRECT,40)

/* This defines a tracepoint at Hoo label. DS:DI points to
   memory that contains a pointer to a structure. We want to
   log the third field in the structure (offset 6 from begin
   of structure). */

TRACE MINOR=0xB2, TP=.Hoo,
      TYPE=(SYS),
      GROUP=FS,
      DESC="(OS) Hoo Pre-Invocation",
      FMT="          Third field in struct is doubleword = %P%D",
      MEM=(RDS+DI,INDIRECT*6,4)

/* This defines a tracepoint at Zoo label. DS:DI points to
   memory that contains a pointer to end of a structure. We
   want to log the last field in the structure(offset -2 from
   end of structure). */

TRACE MINOR=0xB3, TP=.Zoo,
      TYPE=(SYS),
      GROUP=FS,
      DESC="(OS) Zoo Pre-Invocation",
      FMT="          Last field in struct is word = %P%W",
      MEM=(RDS+DI,INDIRECT*-2,2)

/* This defines a tracepoint at procedure CheckIt. This
   is a C routine compiled with debug information. The
   data to log is an ASCIIIZ string called NameIt. */

TRACE MINOR=0xB3, TP=.CheckIt,
      TYPE=(PRE),
      GROUP=FS,
      DESC="(OS) CheckIt Pre-Invocation",
      FMT="          NameIt = %P%S",
      ASCIIIZ=(.NameIt,DIRECT,64)

/* This defines a tracepoint at the return point of the
   procedure CheckIt, a C routine compiled with debug.
   Status_Rec is a record variable. We want to log the
   age field (four bytes from the begin of Status_Rec),
   the name (six bytes from Status_Rec that points to
   an ASCIIIZ string), the age of the next Status_Rec
   (a pointer to the next Status_Rec is ten bytes from
   the begin of Status_Rec, the age is four bytes from
   the begin of the next Status_Rec). */

TRACE MINOR=0x80B3, TP=.CheckIt,RETEP,
      TYPE=(POST),
      GROUP=FS,
      DESC="(OS) CheckIt Post-Invocation",
      FMT="          Status_Rec.age = %P%W",
      FMT="          Status_Rec.name = %P%S",
      FMT="          Status_Rec.next->age = %P%W",
      MEM=(.Status_Rec+4,DIRECT,2),
      ASCIIIZ=(.Status_Rec+6,INDIRECT,64),
      MEM=(.Status_Rec+10,INDIRECT*4,2)

/* This defines a tracepoint at line 58 in the source
   file check.c Debug info is needed to use this
   type of tracepoint. v_ptr is a pointer to a variable
   sized record. The length is 4 bytes past the
   beginning of the record. Log that record. */

```

```

TRACE    MINOR=0x71B4, TP=@check.c,58,
        TYPE=(SYS),
        GROUP=FS,
        DESC="(OS) CheckIt   before allocation",
        FMT="                Variant Record = %P%W%D%U",
        LEN=(v_ptr,INDIRECT**+4),
        MEM=(.v_ptr,INDIRECT,LEN)

        /* This does not define a tracepoint, it only defines a
           trace formatting string for minor code 181 (B5 hex). */

TRACE    MINOR=0xB5, TP=@STATIC,
        DESC="(OS) StaticProcedure Pre-Invocation",
        FMT="                DI = %W FLAGS = %W"

        /* This defines a tracepoint at routine LookUp, but no
           data is to be logged, only the DESC will show up
           in the Trace log when the tracepoint is formatted. */

TRACE    MINOR=0xB6, TP=.LookUp,
        TYPE=(SYS),
        GROUP=FS,
        DESC="(APP) LookUp   Pre-Invocation",

```

-----

## TSF Using 32-bit Addressing

```

; Trace source file for the NEW dynalink.  Compiled with 32-bit offsets.

MODNAME=NEWCALLS.DLL
MAJOR=241
MAXDATALEN=200
; We will want to trace up to 200 bytes in any one trace call.

TYPELIST NAME=API,ID=08,
        NAME=SYS,ID=04,
        NAME=PRE,ID=02,
        NAME=POST,ID=64

GROUPLIST NAME=MEM,ID=1,
        NAME=FS,ID=3

        /* The following tracepoint does not need debug info,
           only a MAP file is necessary with label NewAllocSeg
           public in it.  The program must be compiled in 32-bit
           mode because flat addressing is used (ASCIIIZ32 instead
           of ASCIIIZ).
           This logs lower word of EAX, the double word of EBX
           and the string at the address specified by ESP with
           offset ESI. */

TRACE    MINOR=45, TP=.NewAllocSeg,
        TYPE=(API,PRE),
        GROUP=MEM,
        DESC="(NEW) NewAllocSeg Pre-Invocation",
        FMT ="                AX = %W ",
        FMT ="                EBX = %F",
        FMT ="                param = %P%S",
        REGS=(AX,EBX),
        ASCIIIZ32=(FESP+ESI,DIRECT,20)

        /* This defines a tracepoint at Foo label.  The ten words

```

```

        to log are found indirectly by using EBP with offset
        EDI. Note that each value logged needs a format control. */

TRACE  MINOR=0xD0, TP=.Foo,
        TYPE=(SYS),
        GROUP=FS,
        DESC="(NEW) Foo Pre-Invocation",
        FMT="                First Five words = %P%W%W%W%W%",
        FMT="                Three words ignored %I6",
        FMT="                Last Two Words = %W%W",
        MEM32=(FEBP+EDI,INDIRECT,20)

/* This defines a tracepoint at Goo label. EAX + EDI points
   to a structure whose second field is a pointer to an
   ASCIIIZ string. The offset from the first field in the
   structure is 4 bytes. Max string size to log is 40 bytes.*/

TRACE  MINOR=0xD1, TP=.Goo,
        TYPE=(SYS),
        GROUP=FS,
        DESC="(NEW) Goo Pre-Invocation",
        FMT="                Second field in struct points to %P%S",
        ASCIIIZ32=(FEAX+EDI+4,INDIRECT,40)

/* This defines a tracepoint at Hoo label. EBP + EDI points
   to memory that contains a pointer to a structure. We want
   to log the third field in the structure (offset 6 from
   begin of structure). */

TRACE  MINOR=0xD2, TP=.Hoo,
        TYPE=(SYS),
        GROUP=FS,
        DESC="(NEW) Hoo Pre-Invocation",
        FMT="                Third field in struct is doubleword = %P%D",
        MEM32=(FEBP+EDI,INDIRECT*+6,4)

/* This defines a tracepoint at Zoo label. EAX + EDI points
   to memory that contains a pointer to end of a structure. We
   want to log the last field in the structure (offset -2 from
   end of structure). */

TRACE  MINOR=0xD3, TP=.Zoo,
        TYPE=(SYS),
        GROUP=FS,
        DESC="(OS) Zoo Pre-Invocation",
        FMT="                Last field in struct is word = %P%W",
        MEM=(FEAX+EDI,INDIRECT*-2,2)

/* This defines a tracepoint at procedure CheckIT. This is
   a C routine compiled with debug information. The
   data to log is an ASCIIIZ string called NameIt. */

TRACE  MINOR=0xD3, TP=.CheckIt,
        TYPE=(PRE),
        GROUP=FS,
        DESC="(NEW) CheckIt Pre-Invocation",
        FMT="                NameIt = %P%S",
        ASCIIIZ32=(.NameIt,DIRECT,64)

/* This defines a tracepoint at the return point of the
   procedure CheckIt, a C routine compiled with debug.
   Status_Rec is a record variable. We want to log the
   age field (four bytes from the begin of Status_Rec)
   the name (six bytes from Status_Rec that points to
   an ASCIIIZ string) and the age of the next Status_Rec
   (a pointer to the next Status_Rec is ten bytes from
   the begin of Status_Rec, the age is four bytes from
   the begin of the next Status_Rec). */

TRACE  MINOR=0x80D3, TP=.CheckIt,RETEP,
        TYPE=(POST),
        GROUP=FS,
        DESC="(NEW) CheckIt Post-Invocation",
        FMT="                Status_Rec.age = %P%W",

```

```

FMT="                Status_Rec.name = %P%S",
FMT="                Status_Rec.next->age = %P%W",
MEM32=(.Status_Rec+4,DIRECT,2),
ASCIIIZ32=(.Status_Rec+6,INDIRECT,64),
MEM32=(.Status_Rec+10,INDIRECT*+4,2)

/* This does not define a tracepoint, it only defines a
   trace formatting string for minor code 223 (DF hex). */

TRACE  MINOR=0xDF, TP=@STATIC,
DESC="(NEW) StaticProcedure Pre-Invocation",
FMT="                DI = %W FLAGS = %W"

/* This defines a tracepoint at routine LookUp, but no
   data is to be logged, only the DESC will show up
   in the Trace log when the tracepoint is formatted.
   LookUp is a C language routine not compiled with
   debug and not declared with Pascal
   calling conventions; the underscore is needed for
   this label. */

TRACE  MINOR=0xE0, TP=._LookUp,
TYPE=(SYS),
GROUP=FS,
DESC="(NEW) LookUp  Pre-Invocation"

```

-----

## TSF Using Mix of 16-bit and 32-bit Addressing

```

; Trace source file for the MIXED dynalink.
; Parts were compiled with 16-bit compiler, some with 32-bit compiler.
; The developer must know how the parameters being sent in are
; to be addressed, whether they are segmented or flat addresses.

MODNAME=MIXCALLS.DLL
MAJOR=250
MAXDATALEN=200
; We will want to trace up to 200 bytes in any one trace call.

TYPELIST NAME=API,ID=08,
NAME=SYS,ID=04,
NAME=PRE,ID=02,
NAME=POST,ID=64

GROUPLIST NAME=MEM,ID=1,
NAME=FS,ID=3

/* The following tracepoint is for the routine MixStub.
   This was compiled using segmented addressing and
   one of the parameters to it is a pointer to a control
   block called mix_ctrl. This pointer, found at SS:SP,
   is a flat address because the routine that sent it was
   compiled with the flat addressing specification.
   This logs the mix_ctrl block for 6 bytes. */

TRACE  MINOR=95, TP=.MixStub,
TYPE=(API,PRE),
GROUP=MEM,
DESC="(OS) MixStub      Pre-Invocation",
FMT="                mix_ctrl = %P%W %W %W",
MEM=(RSS+SP,IF,6)      /* is an indirect flat address */

/* The following is for the routine FlatStub. This was

```



```

compiled using 32-bit flat addresses. A parameter to
flatstub is a pointer called p_seg_info. This
pointer is a segmented address because the routine
calling flatstub was compiled using 16-bit segmented
addressing. Log where p_seg_info points for 2 bytes. */

TRACE  MINOR=0xf0, TP=.FlatStub,
        TYPE=(SYS),
        GROUP=FS,
        DESC="(OS) FlatStub Pre-Invocation",
        FMT="                seg_info = %P%W",
        MEM32=(.p_seg_info,IS,2) /* value p_seg_info is a 16-bit */
                                /* segmented address */

```

-----

## Trace Customizer Messages

The messages generated by the Trace Customizer are given below. In addition to the message itself, the source line in error is displayed.

Errors in the FATAL and SEVERE category will cause the compiler to abort immediately.

ERROR messages will normally cause a tracepoint definition to be discarded and processing continues with the next definition.

WARNING messages will allow a valid tracepoint definition to be produced, and the results will normally be as expected.

```
TRCUST(n) severity: message_text
```

where:

**n** is the line number of the tsf in error or (1) if the command syntax is in error.

**severity** is the message severity and may take one of the following values:

- **FATAL**
- **SEVERE**
- **ERROR**
- **WARNING**

**message\_text** is the text of the message.

-----

## Fatal Messages

Msg No.	Message Text
1	TSF file not specified
2	file not found or access denied : %s

3	cannot open file : %s
4	error accessing file : %s
5	error writing to file : %s
6	unable to allocate more memory
7	too many tracepoints in file
8	error reading file: %s, Rc = %s
9	changing file pointer for: %s, Rc = %s
10	unknown EXE header type for: %s
11	invalid path specified in combine file
12	max TFF files to combine is 50
13	all TFFs not have same major code, file: %s
14	invalid MAP file extension given in: %s
15	TCF file not specified
16	filename to long: %s
17	token in TSF file exceeds %s bytes
18	invalid page length, %s bytes specified"
19	error detected unpacking page
20	module does not contain code - no page map

-----

## Severe Messages

Msg No.	Message Text
33	module name not specified
34	premature end of file encountered
35	syntax error : missing '%s' before '%s'
36	new line in literal
37	NULL in literal
38	keyword '%s' expected, '%s' found
39	symbolic info not given for %s
40	MAJOR redefinition
41	TDFID redefinition
42	MAXDATALENGTH redefinition
43	line too long in input file: %s
44	Invalid RETEP criteria specified

---

## Error Messages

Msg No.	Message Text
65	number expected, '%s' found
66	unexpected: %s, ignored
67	minor code not specified
68	minor code out of range
69	TYPELIST redefinition, ignored
70	GROUPLIST redefinition, ignored
71	TP redefinition, tracepoint ignored
72	MINOR redefinition, tracepoint ignored
73	OPCODE redefinition, tracepoint ignored
74	syntax error: missing '%s' before '%s'
75	opcode: %s out of range
76	opcode at TP address cannot be traced
77	opcode mismatch at address to apply TP
78	register expected, '%s' found
79	symbol not found: %s
80	address not found
81	segment register expected, '%s' found
82	trace record incomplete, '%s' required
83	RPN command record exceeds 255 bytes
84	invalid parameter: '%s', ignored
85	invalid ID: %s, ignored
86	group/type redefinition: %s, ignored
87	typeid redefinition: %s, ignored
88	groupid redefinition: %s, ignored
89	invalid address specified: %s
90	line number past end of code for file %s
91	Debug info does not exist for: %s
92	line number missing or invalid: %s
93	filename %s not found in Debug info
94	duplicate minor code = %s, ignored
95	duplicate minor code = %s in file %s, ignored

96	variable LEN parameter not preceding
97	RPN stack limit of 16 exceeded
98	invalid flat register specified: %s
99	total FMT format specs above 4096 bytes
100	zero length specified, tracepoint ignored
101	ORBIT redefinition, tracepoint ignored
102	invalid ORBIT value, tracepoint ignored
103	opcode defined after ABORT
104	opcode defined after REMOVE
105	duplicate TP address, ignored
106	/D not allowed with /C, ignored
107	unable to locate end of %s
108	HLL Debug Info not in order - forcing /NODE
109	Ignoring unsupported line numbers, version %s%s
110	Unrecognised line entry type %s. Line numbers ignored

-----

## Warning Messages

Msg No.	Message Text
129	MAXDATALENGTH out of range, 512 used
130	typename unknown: %s, ignored
131	groupname unknown: %s, ignored
132	file: %s, extension invalid, using: %s
133	'%s' expected before '%s', one assumed
134	too many %s, first 16 types, 48 groups used
135	name too long: %s, first 8 characters used
136	linenum in file: %s not found, using #%s
137	bad object number: %s used for file %s
138	offset %s is invalid for object number %s
139	page tracepoint to be applied at not valid
140	MAXDATALENGTH to log could be exceeded
141	MAJOR out of range, 1 used
142	TDFID out of range, 0 used
143	index too large, high word ignored
144	unable to determine return point for %s

-----

## Using the MAKETSF Utility

The purpose of MAKETSF is to extract dynamic trace definitions imbedded in C or ASM source file to which they relate. MAKETSF will also substitute line number information into those trace definitions that are specified by line number and source file reference.

For example:

```
TRACE TP=@myprog.c,1234
```

This specifies a tracepoint at location corresponding to line 1234 in module whose source is "myprog.c".

The problem with this type of specification is that line number reference will need to be updated whenever the source is changed.

MAKETSF allows the trace definitions to be imbedded in the source as comments but in extraction will generate the correct line number information. It does this by detecting the string "**TP=**@" or "**TP=**@" and then inserting the line number specification. If the **TP=** keyword explicitly specifies an address expression then the trace definition is extracted without modification.

To use line number references modules must be compiled and linked with symbolic debugging information. For CSET2 and VisualAge the compile option is /TI, and the LINK386 option is /DE.

### Note:

Optimised code may not place tracepoints in the desired location.

For example, in a C program

```
/**DT here is the header
 * MODNAME=my.exe
 * MAJOR=256
 */

void myproc(char * parm1) {
int result;

.
.
.

/**DT tracepoint definition
 * TRACE TP=@,
 * DESC="a tracepoint",
 * MEM32=(.parm1,I,4),
 * MEM32=(.result,D,4),
 * FMT="parm1=%p result=%f"
 */
```

Similarly in an assembler program:

```
;**DT here is the header
; MODNAME=my.exe
```

```

; MAJOR=256

parml DD ?

myproc proc
    push ebp
    move ebp,esp
.
.
.

;***DT tracepoint definition
; TRACE TP=@,
; DESC="a tracepoint",
; MEM32=(.parml,D,4),
; FMT="parml=%p%f"

```

#### Notes:

The comment block must begin in column 1 with either ;\*\*\*DT or /\*\*\*DT.

If the C form is used then an \* must appear in column 2 of each line. A \*/ in column 2 ends the comment block.

If the ASM form is used then a ; is required in every column.

If TP=@ is coded then MAPTSF will insert the file name and line number corresponding to the trace definition.

MAKETSF will respond to the \$include <filename> directive if coded in column 1. This is intended for use where multiple source modules comprise a single load module. A list of component source files can be coded in a single file using include statements then given to MAKETSF for processing. For example:

```

A sample include file for MAKETSF

$include myprog.c
$include ../asm/myproc.asm
$include ../sub/subr.c This is a comment

```

MAKETSF will ignore any line that is not part of a ;\*\*\*DT or /\*\*\*DT comment block or a \$include statement.

TSF output from MAKETSF is written to the output file if specified, otherwise to STDOUT.

Messages are written to STDERR.

-----

## MAKETSF Syntax and command line options

The syntax for MAKTSF is:

```
MAKETSF <options> input
```

There are five switch options that MAKETSF supports:

**-a**

append output to output file. The default is to replace the output file.

**-b<path>**

specifies the base directory path (or drive) from which input files specifications are assumed to be relative. If not specified then the current directory is assumed to be the base path.

**Note:**

Note: the base path is not included in the file specification of the TP= keyword, whether or not -p is specified.

**-n**

no includes.

This forces MAKETSF to ignore \$include directives.

**-o<output>**

specifies the output file. The default is to direct output to STDOUT.

**-p**

include source file path information.

This forces MAKETSF to retain the full file specification from \$include statements or the command line input file in the TP= keyword. The default is to strip path information.

**-v**

verbose mode.

This gives information about number of files read and blocks processed.

-----

## Using the DEBDEL Utility

The DEBDEL utility may be used to remove debugging information from a load module after TRCUST has generated the TDF. This avoids the need to re-link edit the load module without /DE option.

Removal of debugging information does not affect the validity of the TDF, however if the source is re-compiled then the TDF must be re-built.

The syntax for DEBDEL is:

```
DEBDEL module_file
```

There is only one parameter: **module\_file**. This is the load module file to be stripped of debugging information.

**Note:**

DEBDEL updates the module in place without first copying it.

-----

## Using the MAPTSF Utility

The MAPTSF utility will generate a TSF from a MAP file. One tracepoint is generated for each public code symbol. Optionally a return tracepoint may be generated for each public code symbol.

The syntax for MAPTSF is as follows:

```
MAPTSF map_file [/MAJOR=major_code]
              [/MODNAME=name]
```

```
[ /MAXDATALENGTH=max_data_length]
[ /MINORSTART=minor_code]
[ /TEMPLATE=template_file]
[ /LOGSTACK=stack_bytes]
[ /EXCLUDE=string[*][,...]]
[ /INCLUDE=string[*][,...]]
[ /REGISTERS=reg[,reg]...]
[ /LOGRETURN]
[ /RETEP]
[ /CASESENSITIVE]
[ /TYPES]
[ /GROUPS=string[,....]]
```

There is only one required parameter: **map\_file**, which specifies the input MAP file. The remaining parameters are optional and have the following meaning:

#### **/MAJOR**

specifies the major code to be used in the MAJOR= statement of the TSF . If omitted, TRCUST will select the default major code of 1 when compiling the TSF.

#### **/MODNAME**

specifies the module name to be used in the MODNAME= statement of the TSF. If omitted, MAPTSF will use the module name that appears in the second line of the MAP file. Note, the MAP file excludes the module extension. TRCUST will assume an extension of .DLL if not specified.

#### **/MAXDATALENGTH**

specifies the MAXDATALENGTH= statement of the TSF. If omitted, TRCUST will assume the default of 512 when compiling the TSF.

#### **/MINORSTART**

specifies the first minor code. Subsequent tracepoints have incremental minor codes. If omitted, the MINOR= statement is not generated. TRCUST will assume an initial minor code of 1 and increment for each tracepoint.

#### **/INCLUDE**

specifies a comma delimited list of case insensitive strings used as inclusion criteria for public symbols. An optional trailing \* signifies a generic match. If both /INCLUDE and /EXCLUDE are specified then the logical OR of their criteria is used for selection. For example:

/I=dos\*,strupr includes all public symbols beginning 'dos' or equal to 'strupr'.

/E=s\* /I=strupr excludes all public symbols beginning 's' except for 'strupr' and includes everything else.

#### **/EXCLUDE**

specifies a comma delimited list of case insensitive strings used as exclusion criteria for public symbols. An optional trailing \* signifies a generic match. If both /INCLUDE and /EXCLUDE are specified then the logical OR of their criteria is used for selection. For example:

/E=\_\*,dos\* excludes all public symbols beginning '\_' or 'dos'.

/E=s\* /I=strupr excludes all public symbols beginning 's' except for 'strupr' and includes everything else.

#### **/CASESENSITIVE**

switch applies to /INCLUDE and /EXCLUDE. If specified then the include and exclude strings will be match on a case-sensitive basis.

#### **/LOGSTACK=n**

specifies the number of bytes of stack to log for entry tracepoints. This causes the following TSF statements to be generated for each entry tracepoint:

for 16-bit code -

```
REGS=( SP,SS) ,
FMT= "Stack pointer SS:SP=%A->" ,
MEM=( RSS+SP,D,n) ,
FMT= " %R%W"
```

for 32-bit code -



```
REGS=( ESP ) ,
FMT="Stack pointer ESP=%F->" ,
MEM32=( FESP,D,n)
FMT=" %R%F"
```

If **/LOGSTACK** is specified without a value then 16 bytes is assumed.

## **/LOGRETURN**

specifies that for each return tracepoint, the return value in AX/EAX should be logged. This causes the following TSF statements to be generated:

for 16-bit code:

```
REGS=( AX )
FMT="Returns (ax) %W"
```

for 32-bit code:

```
REGS=( EAX )
FMT="Returns (eax) %F"
```

## **/REGISTERS**

specifies one or more processor registers to be logged, each separated by a comma. The following register mnemonics are supported:

```
AX,BX,CX,DX,CS,DS,ES,FS,GS,IP,SI,DI,SP,BP,FLAGS,
EAX,EBX,ECX,EDX,EIP,ESI,EDI,ESP,EBP,EFLAGS
```

/REGS may be used as a synonym for /REGISTERS.

## **/RETEP**

specifies that for each public entry-point in the MAP file, a return tracepoint should be generated using the RETEP parameter of the TRACE statement in the TSF.

## **/TYPES**

specifies that generated tracepoints are to be assigned one or more of the following pre-defined types:

PUB	Public routines - names the begin upper case (ignoring leading underscores)
PRIV	Private routines - names the begin lower case (ignoring leading underscores)
PRE	Entry tracepoint.
POST	Exit tracepoint.

## **/GROUPS**

Requests that each of the strings listed be used to define a group. Tracepoints are assigned to a group according to whether a group name matches the beginning of the tracepoint name, ignoring case and leading underscore characters

## **/TEMPLATE**

specifies a file where up to four template tracepoint definitions may be specified, one for each of the following categories:

- 16-bit entry points
- 16-bit return points
- 32-bit entry points
- 32-bit return points

The definitions are in a shortened form of the TRCUST TRACE statement syntax. They are appended to each tracepoint of the category to which they apply. All parameters other than MINOR and DESC are permissible. TP and RETEP are specified as follows:

TP=@16	signifies a 16-bit entry-point
TP=@16,RETEP	signifies a 16-bit return-point
TP=@32	signifies a 32-bit entry-point
TP=@32,RETEP	signifies a 32-bit return-point

Only TP and RETEP may appear on the same line as the TRACE keyword.

For example:

```
TRACE TP=@16
MEM=(SS:BP+8,I,0x10)
FMT="16-bytes of parameter 1: %R%W"
```

will append

```
MEM=(SS:BP+8,I,0x10)
FMT="16-bytes of parameter 1: %R%W"
```

to every 16-bit entry tracepoint definition.

-----

## System Tracepoints Reference

This chapter documents all static and dynamic tracepoints that are defined in the base OS/2 system code.

Tracepoint definitions are listed in the section: [Trace Event Major and Minor Code Assignments](#). Each tracepoint listing is divided into the following six fields:

Event Description	A description of the event that is being captured (for example, DosOpen Pre-invocation)
Tracepoint	The public symbol or relative location at which the tracepoint is defined.
Minor Code	The minor trace code that is associated with the event
Type	A qualifier that can be used within the Trace control utility to enable or disable sets of dynamic tracepoints with a single command. <a href="#">Type Qualifiers</a> are discussed in more detail in a following section.
Group	A second type of qualifier that can be used within the Trace control utility to enable or disable sets of dynamic tracepoints with a single command. <a href="#">Group Qualifiers</a> are discussed in more detail in a following section.
Parameters	The data that is logged by the tracepoint. <a href="#">Parameter Notation</a> is discussed in more detail in a following section.

For information on defining tracepoints consult the following:

- The dynamic trace customiser utility [TRCUST](#)

- The [DosSysTrace](#) API.

---

## Type Qualifiers

The following Type identifiers are currently defined. They are intended to simplify the identification of components and to aid in selecting events to be traced:

PRE	A pre-invocation event
POST	A post-invocation event
API	An external interface
INT	An internal interface

Typically, pairs of Type identifiers are used (for example, PRE and API).

---

## Group Qualifiers

The following Group identifiers are currently defined. They are intended to simplify the identification of components and to aid in selecting events to be traced:

EXMG	Exception Management
FS	File System
IO	Device I/O
KBD	Keyboard
LDR	Loader
LNK	Environment
LOCK	SMP Spink Locks
MOU	Mouse
MSP	Memory Suballocation
NLS	National Language Services
PIP	Pipes
QUE	Queues
SEL	Selector Management
SEM	Semaphores
SIG	Signals
TIM	Timers
TK	Tasking
UT	UnitThunk Processing
VIO	Video I/O
VM	Virtual Memory Management

---

## Parameter Format Notation

The Parameter portion of a tracepoint listing typically includes both fixed descriptive ASCII strings and variable format descriptors. The following list enumerates the classes of format descriptors that are encountered in the tracepoint listings:

%A	Output a segmented (16:16) address (32 bits)
%B	Output a byte (8 bits) of data
%D	Output A double word (32 bits) of data
%F	Output a flat (0:32) address (32 bits)
%Q	Output a quad word (64 bits) of data
%S	Output an ASCIIZ string (which is "n" bytes terminated by 00h)
%U	Format the remainder of the trace record as a sequence of bytes
%W	Output a word (16 bits) of data
%X	Output the major event code
%Y	Output the minor event code

A number preceding one of the above specifies a sequence of that type.

## DosXxxx API Pre-invocation Tracepoints.

From OS/2 Warp V3.0 fix pack 30 and OS/2 Warp V4.0 fix pack 10 all system API pre-invocation tracepoints have been updated to log the caller's return address. In most cases the meaning of this information is self-evident and the original tracepoint definition has been updated to include a line that displays as follows:

```
32-bit API:
    Return address = %F

16-bit API:
    Return address = %W:%W
```

Special consideration needs to be given to APIs involving the KERNEL, SESMGR and QUECALLS. These are APIs the use the **Dos** prefix in their names and my indirect though DOSCALL1.

The following topics describe each of these cases:

- [Tracing Kernel API Return Information](#)
- [Tracing Session Manager API return information](#)
- [Tracing Queue Calls API return information](#)

## Tracing Kernel API Return Information

### Kernel APIs

There three schemes that operate when kernel API is called:

```
Direct call:
    APPL          > KERNEL
    <

Indirect call, direct return:
    APPL          > DOSCALL1      > KERNEL
    <

Indirect call, indirect return:
    APPL          > DOSCALL1      > KERNEL
    <          DOSCALL1 <
```

### Direct Calls:

Kernel APIs are entered directly from the application. Return information is logged in the following format:

32-bit API:

Return EIP=%F CS=%W

16-bit API:

Return IP=%W CS=%W

CS, EIP and IP refer to the return address in the application.

Use TRACE ON KERNEL(...) to trace these APIs.

Indirect call, direct return:

32-bit Kernel APIs are preprocessed by DOSCALL1 but are returned to directly from the KERNEL. Return information is logged in the following format:

32-bit API:

Return EIP=%F CS=%W, Thunk EIP=%F

Thunk EIP refers to the return address in to the application. Return EIP and CS refer to the return back to DOSCALL1. In most cases the pre-invocation tracepoint is recoded after the kernel has updated the Return EIP with the Thunk EIP value and they will have the same value.

Use TRACE ON KERNEL(...) to trace these APIs.

Indirect call, Indirect return:

32-bit APIs are pre- and post-processed by DOSCALL1. Parameters are logged by the kernel tracepoint, but the return address will only show the direct return back to DOSCALL1. For each API of this form a pre-invocation API in DOSCALL1 has been defined that logs just the return address back to the application. For example, the following shows DosSleep entry to the Kernel, preceded by Dos32Sleep entry to DOSCALL1:

```
(OS) DosSleep Pre-Invocation
Event [10] Major [5/0x0005] Minor [307/0x0133] PID [38/0x0026] Length [18] Time [18:07:32]
Return EIP=0000C361 CS=DFD7
Timeout Interval = 0000 0000

(OS) Dos32Sleep Pre-Invocation
Event [11] Major [16/0x0010] Minor [267/0x010b] PID [38/0x0026] Length [7] Time [18:07:32]
Return address = 1BDFAA63
```

For a list of kernel APIs indirected through DOSCALL1 see: [Kernel API Tracepoints Indirected Via DOSCALL1](#).

## Tracing Session Manager API return information

### SESMGR APIs

SESMGR APIs are entered either directly or indirectly via a SESMGR thunking layer as shown below:

Direct call:

```
APPL                                     > SESMGR
<
```

Thunked call:

```
APPL      > SESMGR Thunk Layer      > SESMGR
          < SESMGR Thunk Layer <
```

Direct Calls:

Used for 16-bit APIs. Use TRACE ON SESMGR(....) for tracing APIs and return information.

Thunked Calls:

Used for 32-bit SESMGR APIs. Trace the 16-bit SESMGR API with the 32-bit SESMGR API to record the return information to the original 32-bit caller. For example, to trace the return information and parameters to **DosStartSession** from a 32-bit caller use: TRACE ON SESMGR(20,34,32788). The following is an example of the traced output:

```
(OS) DosStartSession Post-Invocation
Event [8] Major [23/0x0017] Minor [32788/0x8014] PID [62/0x003e] Length [12] Time [10:44:22]
      New Process Id      = 0000
      New Session         = f0000
      Return Code         = 0000

(OS) DosStartSession Pre-Invocation
Event [16] Major [23/0x0017] Minor [20/0x0014] PID [62/0x003e] Length [60] Time [10:44:22]
Return address = DFD7:BD0C
      Start Data          = 0032 0000 0000 0000 62D4 004F 2B54 004F CA10 045F 0000 0000 00
0000 0000 0000 0000 0000 0000 0000 0000 0000

(OS) Dos32StartSession Pre-Invocation
Event [17] Major [23/0x0017] Minor [34/0x0022] PID [62/0x003e] Length [7] Time [10:44:22]
Return address = 0002E7E2
```

In this example the return address to the 32-bit application is given in minor code 34.

For a list of indirected SESMR APIs see: [Indirected Session Manager API Tracepoints](#).

-----

## Tracing Queue Calls API return information

### QUECALLS APIs

QUECALLS APIs are entered indirectly via DOSCALL1 as shown below:

```
APPL      > DOSCALL1      > QUECALLS
          < DOSCALL1 <
```

To trace QUECALLS API parameters and results use TRACE ON QUECALLS(.....). To trace the API return addresses use TRACE ON DOSCALL1(.....).

**Note:** The QUECALLS entry points in DOSCALL1 have been grouped using group id **QUE**. All the API return addresses may be traced by specifying TRACE ON DOSCALL1(QUE).

The following is an example of QUECALLS traced output activated using

```
TRACE ON DOSCALL1(QUE)
TRACE ON QUECALLS

(OS) DosReadQueue Pre-Invocation
Event [1] Major [22/0x0016] Minor [7/0x0007] PID [109/0x006d] Length [21] Time [10:58:14.97]
Handle=0014 Element Code=0000
Semaphore Handle=0027 75E4 No Wait Flag=00

(OS) Dos16ReadQueue Pre-Invocation
Event [2] Major [16/0x0010] Minor [360/0x0168] PID [109/0x006d] Length [7] Time [10:58:14.97]
Return address = 000F:47EC
```

In this example the return address to the 16-bit caller is given in minor code 360 of DOSCALL1.  
For a list of indirected QUECALLS tracepoits see: [QUECALLS API Tracepoints Indirected Via DOSCALL1](#).

# Trace Event Major and Minor Code Assignments

This section lists the overall assignment of major codes for the kernel and related subsystems supported by the trace facility.

Usage	Major Code
System Trace internally generated events	00
Dekko performance tracepoints	01
Reserved	02
<a href="#">Machine Exceptions</a>	03
<a href="#">Hardware Interrupts</a>	04
<a href="#">Kernel Services (KERNEL)</a>	05
<a href="#">Device Helper Services</a>	06
<a href="#">Miscellaneous Kernel Events</a>	07
<a href="#">Resource Manager Events (RESOURCE.SYS)</a>	08
Reserved	09 - 15
<a href="#">Kernel Services (DOSCALL1.DLL)</a>	16
<a href="#">Monitor Call Services (MONCALLS.DLL)</a>	16
Reserved	17 - 21
<a href="#">Queue Services (QUECALLS.DLL)</a>	22
<a href="#">Session Manager Services (SESMGR.DLL)</a>	23
<a href="#">Character I/O Services (OS2CHAR.DLL)</a>	24
Reserved	25 - 108
<a href="#">Multi-Media Extensions</a>	109
Reserved	110 - 137

DCAF (Distributed Console Access Facility)	138
Reserved	139 - 150
IBM LAN Manager (RTP)	151
EE LAN Requestor	152 - 159
Communications Manager APPC	160
Communications Manager 3270 Emulator	161
Communications Manager Async	162
Communications Manager SRPI/DLC	163
Communications Manager	164 - 167
Reserved	168
Communications Manager	169
Reserved	170 - 171
Communications Manager	172 - 175
DataBase Manager	176 - 191
PM Shell API (PMSHAPI.DLL)	192
Reserved	193
PM Window Management (PMWIN.DLL)	194
PM Graphics Engine (PMGRE.DLL)	195
PM Picture Interchange (PMPIC.DLL)	196
PM Graphics Programming Interfaces (PMGPI.DLL)	197
PM Print Spooler (PMSPL.DLL)	198
199 - 212	Reserved
Transmission Network Manager	212
Reserved	213
DBM Archival Logging	222 - 225
DBM Query Manager	226 - 235
User Profile Management	236 - 237
238 - 241	Reserved
LAN Station Manager	242
Reserved	243
TCP/IP	244
Available for Temporary Application Use	245 - 255
Kernel Device Manager Internal Tracepoints	256
Kernel File System Manager Internal Tracepoints	257
Kernel Virtual DOS Machine Manager Internal Tracepoints	258
Kernel Module Loader and Manager Internal Tracepoints	259



Kernel Page Manager Internal Tracepoints	260
Kernel Scheduler and Task Manager Internal Tracepoints	261
Kernel Swap File Manager Internal Tracepoints	262
Kernel Selector Manager Internal Tracepoints	263
Kernel Virtual Memory Manager Internal Tracepoints	264
Kernel Semaphore Manager Internal Tracepoints	265
Kernel Timer Manager Internal Tracepoints	266
Reserved	247 - 279
HPFS Internal Tracepoints	280
CDFS Internal Tracepoints	281
Reserved	282 - 299
PMVDMP Interfaces	300
PMVIOP Interfaces	301
PMWP Interfaces	302
JFS Internal Tracepoints	303
TRCUST Internal Tracepoints	304
Reserved	305 - 385
HPFS386 Internal Tracepoints	386
Reserved	387 - 0x7fff
Available for User and Vendor Use	0x8000 - 0xffff

Trace Event Major Code Assignments

# Machine Exceptions Trace Events

The tracepoints for the Machine Exceptions major code are identified in the following table. These tracepoints are static tracepoints. They are compiled with the code.

[Trace events for EXCEPT Major Code: 0X0003, sorted by minor code.](#)  
[Trace events for EXCEPT Major Code: 0X0003 ,sorted by tracepoint.](#)

## Trace Events for EXCEPT Major Code: 0X0003, Sorted by Minor Code

00001 (0X0001) (OS) Machine Exception 00 - Divide by 0  
00002 (0X0002) (OS) Machine Exception 01 - Single Step  
00003 (0X0003) (OS) Machine Exception 02 - NMI  
00004 (0X0004) (OS) Machine Exception 03 - Breakpoint  
00005 (0X0005) (OS) Machine Exception 04 - Overflow  
00006 (0X0006) (OS) Machine Exception 05 - Bounds Check  
00007 (0X0007) (OS) Machine Exception 06 - Invalid Opcode  
00008 (0X0008) (OS) Machine Exception 07 - Coprocessor Not Available  
00009 (0X0009) (OS) Machine Exception 08 - Double Fault  
00010 (0X000A) (OS) Machine Exception 09 - Reserved  
00011 (0X000B) (OS) Machine Exception 0a - Invalid TSS  
00012 (0X000C) (OS) Machine Exception 0b - Segment Not Present  
00013 (0X000D) (OS) Machine Exception 0c - Stack Exception  
00014 (0X000E) (OS) Machine Exception 0d - General Protection  
00015 (0X000F) (OS) Machine Exception 0e - Page Fault

---

## Trace Events for EXCEPT Major Code: 0X0003, Sorted by Tracepoint

(OS) Machine Exception 00 - Divide by 0 00001 (0X0001)  
(OS) Machine Exception 01 - Single Step 00002 (0X0002)  
(OS) Machine Exception 02 - NMI 00003 (0X0003)  
(OS) Machine Exception 03 - Breakpoint 00004 (0X0004)  
(OS) Machine Exception 04 - Overflow 00005 (0X0005)  
(OS) Machine Exception 05 - Bounds Check 00006 (0X0006)  
(OS) Machine Exception 06 - Invalid Opcode 00007 (0X0007)  
(OS) Machine Exception 07 - Coprocessor Not Available 00008 (0X0008)  
(OS) Machine Exception 08 - Double Fault 00009 (0X0009)  
(OS) Machine Exception 09 - Reserved 00010 (0X000A)  
(OS) Machine Exception 0a - Invalid TSS 00011 (0X000B)  
(OS) Machine Exception 0b - Segment Not Present 00012 (0X000C)  
(OS) Machine Exception 0c - Stack Exception 00013 (0X000D)  
(OS) Machine Exception 0d - General Protection 00014 (0X000E)  
(OS) Machine Exception 0e - Page Fault 00015 (0X000F)

---

## EXCEPT Major Code: 0X0003 Minor code: 1 (0X0001)

### Description

(OS) Machine Exception 00 - Divide by 0

### Tracepoint

Static tracepoint in EXCEPT.

### Minor code

1 (0X0001)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 2 (0X0002)

<u>Description</u>	(OS) Machine Exception 01 - Single Step
<u>Tracepoint</u>	Static tracepoint in EXCEPT.
<u>Minor code</u>	2 (0X0002)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 3 (0X0003)

<u>Description</u>	(OS) Machine Exception 02 - NMI
<u>Tracepoint</u>	Static tracepoint in EXCEPT.
<u>Minor code</u>	3 (0X0003)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 4 (0X0004)

<u>Description</u>	(OS) Machine Exception 03 - Breakpoint
<u>Tracepoint</u>	Static tracepoint in EXCEPT.

**Minor code**

4 (0X0004)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 5 (0X0005)

**Description**

(OS) Machine Exception 04 - Overflow

**Tracepoint**

Static tracepoint in EXCEPT.

**Minor code**

5 (0X0005)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 6 (0X0006)

**Description**

(OS) Machine Exception 05 - Bounds Check

**Tracepoint**

Static tracepoint in EXCEPT.

**Minor code**

6 (0X0006)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 7 (0X0007)

<u>Description</u>	(OS) Machine Exception 06 - Invalid Opcode
<u>Tracepoint</u>	Static tracepoint in EXCEPT.
<u>Minor code</u>	7 (0X0007)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 8 (0X0008)

<u>Description</u>	(OS) Machine Exception 07 - Coprocessor Not Available
<u>Tracepoint</u>	Static tracepoint in EXCEPT.
<u>Minor code</u>	8 (0X0008)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 9 (0X0009)

<u>Description</u>	(OS) Machine Exception 08 - Double Fault
<u>Tracepoint</u>	Static tracepoint in EXCEPT.

**Minor code** 9 (0X0009)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 10 (0X000A)

**Description** (OS) Machine Exception 09 - Reserved

**Tracepoint** Static tracepoint in EXCEPT.

**Minor code** 10 (0X000A)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 11 (0X000B)

**Description** (OS) Machine Exception 0a - Invalid TSS

**Tracepoint** Static tracepoint in EXCEPT.

**Minor code** 11 (0X000B)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 12 (0X000C)

<u>Description</u>	(OS) Machine Exception 0b - Segment Not Present
<u>Tracepoint</u>	Static tracepoint in EXCEPT.
<u>Minor code</u>	12 (0X000C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 13 (0X000D)

<u>Description</u>	(OS) Machine Exception 0c - Stack Exception
<u>Tracepoint</u>	Static tracepoint in EXCEPT.
<u>Minor code</u>	13 (0X000D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Error Code = %D, CS:EIP = %Q, EFlags = %D

-----

## EXCEPT Major Code: 0X0003 Minor code: 14 (0X000E)

<u>Description</u>	(OS) Machine Exception 0d - General Protection
<u>Tracepoint</u>	Static tracepoint in EXCEPT.

<u>Minor code</u>	14 (0X000E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Error Code = %D, CS:EIP = %Q, EFlags = %D

## EXCEPT Major Code: 0X0003 Minor code: 15 (0X000F)

<u>Description</u>	(OS) Machine Exception 0e - Page Fault
<u>Tracepoint</u>	Static tracepoint in EXCEPT.
<u>Minor code</u>	15 (0X000F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Error Code = %D, CS:EIP = %Q, EFlags = %D
	Page Fault Linear Address (CR2) = %F

## Hardware Interrupts Trace Events

The tracepoints for the Hardware Interrupts major code are identified in the following tables. These tracepoints are static tracepoints. They are compiled with the code.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for INT Major Code: 0X0004, sorted by minor code.](#)  
[Trace events for INT Major Code: 0X0004 ,sorted by tracepoint.](#)

## Trace Events for INT Major Code: 0X0004, Sorted by Minor Code



00001 (0X0001) (OS) Hardware Interrupt Pre-Invocation  
00002 (0X0002) (OS) Hardware Interrupt Pre-Invocation  
00003 (0X0003) (OS) Hardware Interrupt Pre-Invocation  
00004 (0X0004) (OS) Hardware Interrupt Pre-Invocation  
00005 (0X0005) (OS) Hardware Interrupt Pre-Invocation  
00006 (0X0006) (OS) Hardware Interrupt Pre-Invocation  
00007 (0X0007) (OS) Hardware Interrupt Pre-Invocation  
00008 (0X0008) (OS) Hardware Interrupt Pre-Invocation  
00009 (0X0009) (OS) Hardware Interrupt Pre-Invocation  
00010 (0X000A) (OS) Hardware Interrupt Pre-Invocation  
00011 (0X000B) (OS) Hardware Interrupt Pre-Invocation  
00012 (0X000C) (OS) Hardware Interrupt Pre-Invocation  
00013 (0X000D) (OS) Hardware Interrupt Pre-Invocation  
00014 (0X000E) (OS) Hardware Interrupt Pre-Invocation  
00015 (0X000F) (OS) Hardware Interrupt Pre-Invocation  
00016 (0X0010) (OS) Hardware Interrupt Pre-Invocation  
00017 (0X0011) (OS) Hardware Interrupt Pre-Invocation  
00018 (0X0012) (OS) Hardware Interrupt Pre-Invocation  
00019 (0X0013) (OS) Hardware Interrupt Pre-Invocation  
00020 (0X0014) (OS) Hardware Interrupt Pre-Invocation  
00021 (0X0015) (OS) Hardware Interrupt Pre-Invocation  
00022 (0X0016) (OS) Hardware Interrupt Pre-Invocation  
00023 (0X0017) (OS) Hardware Interrupt Pre-Invocation  
00024 (0X0018) (OS) Hardware Interrupt Pre-Invocation  
00025 (0X0019) (OS) Hardware Interrupt Pre-Invocation  
00026 (0X001A) (OS) Hardware Interrupt Pre-Invocation  
00027 (0X001B) (OS) Hardware Interrupt Pre-Invocation  
00028 (0X001C) (OS) Hardware Interrupt Pre-Invocation  
00029 (0X001D) (OS) Hardware Interrupt Pre-Invocation  
00030 (0X001E) (OS) Hardware Interrupt Pre-Invocation  
00031 (0X001F) (OS) Hardware Interrupt Pre-Invocation  
00032 (0X0020) (OS) Hardware Interrupt Pre-Invocation  
00129 (0X0081) (OS) Hardware Interrupt Post-Invocation  
00130 (0X0082) (OS) Hardware Interrupt Post-Invocation  
00131 (0X0083) (OS) Hardware Interrupt Post-Invocation  
00132 (0X0084) (OS) Hardware Interrupt Post-Invocation  
00133 (0X0085) (OS) Hardware Interrupt Post-Invocation  
00134 (0X0086) (OS) Hardware Interrupt Post-Invocation  
00135 (0X0087) (OS) Hardware Interrupt Post-Invocation  
00136 (0X0088) (OS) Hardware Interrupt Post-Invocation  
00137 (0X0089) (OS) Hardware Interrupt Post-Invocation  
00138 (0X008A) (OS) Hardware Interrupt Post-Invocation  
00139 (0X008B) (OS) Hardware Interrupt Post-Invocation  
00140 (0X008C) (OS) Hardware Interrupt Post-Invocation  
00141 (0X008D) (OS) Hardware Interrupt Post-Invocation  
00142 (0X008E) (OS) Hardware Interrupt Post-Invocation  
00143 (0X008F) (OS) Hardware Interrupt Post-Invocation  
00144 (0X0090) (OS) Hardware Interrupt Post-Invocation  
00145 (0X0091) (OS) Hardware Interrupt Post-Invocation  
00146 (0X0092) (OS) Hardware Interrupt Post-Invocation  
00147 (0X0093) (OS) Hardware Interrupt Post-Invocation  
00148 (0X0094) (OS) Hardware Interrupt Post-Invocation  
00149 (0X0095) (OS) Hardware Interrupt Post-Invocation  
00150 (0X0096) (OS) Hardware Interrupt Post-Invocation  
00151 (0X0097) (OS) Hardware Interrupt Post-Invocation  
00152 (0X0098) (OS) Hardware Interrupt Post-Invocation  
00153 (0X0099) (OS) Hardware Interrupt Post-Invocation  
00154 (0X009A) (OS) Hardware Interrupt Post-Invocation  
00155 (0X009B) (OS) Hardware Interrupt Post-Invocation  
00156 (0X009C) (OS) Hardware Interrupt Post-Invocation  
00157 (0X009D) (OS) Hardware Interrupt Post-Invocation  
00158 (0X009E) (OS) Hardware Interrupt Post-Invocation  
00159 (0X009F) (OS) Hardware Interrupt Post-Invocation  
00160 (0X00A0) (OS) Hardware Interrupt Post-Invocation

-----

Trace Events for INT Major Code: 0X0004, Sorted by

# Tracepoint

[illegible]

---

# INT Major Code: 0X0004 Minor Code: 1 (0X0001)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	1 (0X0001)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Interrupt Level=0 (Timer)

# INT Major Code: 0X0004 Minor Code: 2 (0X0002)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	2 (0X0002)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Interrupt Level=1 (Keyboard)

# INT Major Code: 0X0004 Minor Code: 3 (0X0003)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	

3 (0X0003)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Interrupt Level=2 (NMI)

-----

## INT Major Code: 0X0004 Minor Code: 4 (0X0004)

**Description**

(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**

Static tracepoint in INT.

**Minor Code**

4 (0X0004)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Interrupt Level=3 (Serial Port 2)

-----

## INT Major Code: 0X0004 Minor Code: 5 (0X0005)

**Description**

(OS) Hardware Interrupt Pre-Invocation

**Tracepoint**

Static tracepoint in INT.

**Minor Code**

5 (0X0005)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Interrupt Level=4 (Serial Port 1)

-----

## INT Major Code: 0X0004 Minor Code: 6 (0X0006)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	6 (0X0006)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=5 (Parallel Port 2)

-----

## INT Major Code: 0X0004 Minor Code: 7 (0X0007)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	7 (0X0007)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=6 (Diskette Controller)

-----

## INT Major Code: 0X0004 Minor Code: 8 (0X0008)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code** 8 (0X0008)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=7 (Parallel Port 1)

-----

INT Major Code: 0X0004 Minor Code: 9 (0X0009)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 9 (0X0009)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=8 (Realtime Clock)

-----

INT Major Code: 0X0004 Minor Code: 10 (0X000A)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 10 (0X000A)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=9

-----

## INT Major Code: 0X0004 Minor Code: 11 (0X000B)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	11 (0X000B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=A

-----

## INT Major Code: 0X0004 Minor Code: 12 (0X000C)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	12 (0X000C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=B

-----

## INT Major Code: 0X0004 Minor Code: 13 (0X000D)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code** 13 (0X000D)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=C

-----

INT Major Code: 0X0004 Minor Code: 14 (0X000E)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 14 (0X000E)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=D (Coprocessor)

-----

INT Major Code: 0X0004 Minor Code: 15 (0X000F)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 15 (0X000F)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=E (Fixed Disk Controller)



-----

## INT Major Code: 0X0004 Minor Code: 16 (0X0010)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	16 (0X0010)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=F

-----

## INT Major Code: 0X0004 Minor Code: 17 (0X0011)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	17 (0X0011)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=10

-----

## INT Major Code: 0X0004 Minor Code: 18 (0X0012)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code** 18 (0X0012)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=11

-----

INT Major Code: 0X0004 Minor Code: 19 (0X0013)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 19 (0X0013)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=12

-----

INT Major Code: 0X0004 Minor Code: 20 (0X0014)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 20 (0X0014)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=13

-----

## INT Major Code: 0X0004 Minor Code: 21 (0X0015)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	21 (0X0015)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=14

-----

## INT Major Code: 0X0004 Minor Code: 22 (0X0016)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	22 (0X0016)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=15

-----

## INT Major Code: 0X0004 Minor Code: 23 (0X0017)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

<b><u>Minor Code</u></b>	23 (0X0017)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	Interrupt Level=16

-----

## INT Major Code: 0X0004 Minor Code: 24 (0X0018)

<b><u>Description</u></b>	(OS) Hardware Interrupt Pre-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in INT.
<b><u>Minor Code</u></b>	24 (0X0018)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	Interrupt Level=17

-----

## INT Major Code: 0X0004 Minor Code: 25 (0X0019)

<b><u>Description</u></b>	(OS) Hardware Interrupt Pre-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in INT.
<b><u>Minor Code</u></b>	25 (0X0019)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	Interrupt Level=18

-----

## INT Major Code: 0X0004 Minor Code: 26 (0X001A)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	26 (0X001A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=19

-----

## INT Major Code: 0X0004 Minor Code: 27 (0X001B)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	27 (0X001B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=1A

-----

## INT Major Code: 0X0004 Minor Code: 28 (0X001C)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code** 28 (0X001C)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=1B

-----

INT Major Code: 0X0004 Minor Code: 29 (0X001D)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 29 (0X001D)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=1C

-----

INT Major Code: 0X0004 Minor Code: 30 (0X001E)

**Description** (OS) Hardware Interrupt Pre-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 30 (0X001E)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Interrupt Level=1D

-----

## INT Major Code: 0X0004 Minor Code: 31 (0X001F)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	31 (0X001F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=1E

-----

## INT Major Code: 0X0004 Minor Code: 32 (0X0020)

<u>Description</u>	(OS) Hardware Interrupt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	32 (0X0020)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interrupt Level=1F

-----

## INT Major Code: 0X0004 Minor Code: 129 (0X0081)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code** 129 (0X0081)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=0 (Timer)

-----

INT Major Code: 0X0004 Minor Code: 130 (0X0082)

**Description** (OS) Hardware Interrupt Post-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 130 (0X0082)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=1 (Keyboard)

-----

INT Major Code: 0X0004 Minor Code: 131 (0X0083)

**Description** (OS) Hardware Interrupt Post-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 131 (0X0083)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=2 (NMI)



-----

## INT Major Code: 0X0004 Minor Code: 132 (0X0084)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	132 (0X0084)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=3 (Serial Port 2)

-----

## INT Major Code: 0X0004 Minor Code: 133 (0X0085)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	133 (0X0085)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=4 (Serial Port 1)

-----

## INT Major Code: 0X0004 Minor Code: 134 (0X0086)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code**

134 (0X0086)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Interrupt Level=5 (Parallel Port 2)

-----

## INT Major Code: 0X0004 Minor Code: 135 (0X0087)

**Description**

(OS) Hardware Interrupt Post-Invocation

**Tracepoint**

Static tracepoint in INT.

**Minor Code**

135 (0X0087)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Interrupt Level=6 (Diskette Controller)

-----

## INT Major Code: 0X0004 Minor Code: 136 (0X0088)

**Description**

(OS) Hardware Interrupt Post-Invocation

**Tracepoint**

Static tracepoint in INT.

**Minor Code**

136 (0X0088)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Interrupt Level=7 (Parallel Port 1)

-----

## INT Major Code: 0X0004 Minor Code: 137 (0X0089)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	137 (0X0089)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=8 (Realtime Clock)

-----

## INT Major Code: 0X0004 Minor Code: 138 (0X008A)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	138 (0X008A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=9

-----

## INT Major Code: 0X0004 Minor Code: 139 (0X008B)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code** 139 (0X008B)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=A

-----

INT Major Code: 0X0004 Minor Code: 140 (0X008C)

**Description** (OS) Hardware Interrupt Post-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 140 (0X008C)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=B

-----

INT Major Code: 0X0004 Minor Code: 141 (0X008D)

**Description** (OS) Hardware Interrupt Post-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 141 (0X008D)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=C

-----

## INT Major Code: 0X0004 Minor Code: 142 (0X008E)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	142 (0X008E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=D (Coprocessor)

-----

## INT Major Code: 0X0004 Minor Code: 143 (0X008F)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	143 (0X008F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=E (Fixed Disk Controller)

-----

## INT Major Code: 0X0004 Minor Code: 144 (0X0090)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

<b><u>Minor Code</u></b>	144 (0X0090)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Interrupt Level=F

-----

## INT Major Code: 0X0004 Minor Code: 145 (0X0091)

<b><u>Description</u></b>	(OS) Hardware Interrupt Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in INT.
<b><u>Minor Code</u></b>	145 (0X0091)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Interrupt Level=10

-----

## INT Major Code: 0X0004 Minor Code: 146 (0X0092)

<b><u>Description</u></b>	(OS) Hardware Interrupt Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in INT.
<b><u>Minor Code</u></b>	146 (0X0092)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Interrupt Level=11

-----

## INT Major Code: 0X0004 Minor Code: 147 (0X0093)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	147 (0X0093)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=12

-----

## INT Major Code: 0X0004 Minor Code: 148 (0X0094)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	148 (0X0094)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=13

-----

## INT Major Code: 0X0004 Minor Code: 149 (0X0095)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

<b><u>Minor Code</u></b>	149 (0X0095)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
	Interrupt Level=14

-----

## INT Major Code: 0X0004 Minor Code: 150 (0X0096)

<b><u>Description</u></b>	(OS) Hardware Interrupt Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in INT.
<b><u>Minor Code</u></b>	150 (0X0096)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
	Interrupt Level=15

-----

## INT Major Code: 0X0004 Minor Code: 151 (0X0097)

<b><u>Description</u></b>	(OS) Hardware Interrupt Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in INT.
<b><u>Minor Code</u></b>	151 (0X0097)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
	Interrupt Level=16



-----

## INT Major Code: 0X0004 Minor Code: 152 (0X0098)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	152 (0X0098)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=17

-----

## INT Major Code: 0X0004 Minor Code: 153 (0X0099)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	153 (0X0099)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=18

-----

## INT Major Code: 0X0004 Minor Code: 154 (0X009A)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

**Minor Code** 154 (0X009A)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=19

-----

INT Major Code: 0X0004 Minor Code: 155 (0X009B)

**Description** (OS) Hardware Interrupt Post-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 155 (0X009B)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=1A

-----

INT Major Code: 0X0004 Minor Code: 156 (0X009C)

**Description** (OS) Hardware Interrupt Post-Invocation

**Tracepoint** Static tracepoint in INT.

**Minor Code** 156 (0X009C)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Interrupt Level=1B

-----

## INT Major Code: 0X0004 Minor Code: 157 (0X009D)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	157 (0X009D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=1C

-----

## INT Major Code: 0X0004 Minor Code: 158 (0X009E)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.
<u>Minor Code</u>	158 (0X009E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Interrupt Level=1D

-----

## INT Major Code: 0X0004 Minor Code: 159 (0X009F)

<u>Description</u>	(OS) Hardware Interrupt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in INT.

<b><u>Minor Code</u></b>	159 (0X009F)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
	Interrupt Level=1E

-----

INT Major Code: 0X0004 Minor Code: 160 (0X00A0)

<b><u>Description</u></b>	(OS) Hardware Interrupt Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in INT.
<b><u>Minor Code</u></b>	160 (0X00A0)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
	Interrupt Level=1F

-----

KERNEL Services Trace Events

The external API tracepoints for the KERNEL major code are identified in the following tables. These tracepoints are dynamic tracepoints.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for OS2KRNL Major Code: 0X0005, sorted by minor code.](#)  
[Trace events for OS2KRNL Major Code: 0X0005 ,sorted by tracepoint.](#)  
[Kernel API Tracepoints Indirected Via DOSCALL1.](#)

-----

Trace Events for OS2KRNL Major Code: 0X0005, Sorted by Minor Code

00001 (0X0001) (OS) DosEnterCritSec Post-Invocation  
00002 (0X0002) (OS) DosExitCritSec Post-Invocation  
00003 (0X0003) (OS) DosHoldSignal Post-Invocation (2)  
00006 (0X0006) (OS) Dos32AliasMem Post-Invocation  
00007 (0X0007) (OS) Dos32AllocMem Post-Invocation  
00008 (0X0008) (OS) Dos32AllocProtectedMem Post-Invocation  
00009 (0X0009) (OS) Dos32AllocSharedMem Post-Invocation  
00010 (0X000A) (OS) Dos32CreateThread Post-Invocation  
00011 (0X000B) (OS) Dos32Debug Dos Post-Invocation  
00012 (0X000C) (OS) Dos32ExitList Dos Post-Invocation  
00013 (0X000D) (OS) Dos32FreeMem Post-Invocation  
00014 (0X000E) (OS) Dos32GetNamedSharedMem Post-Invocation  
00015 (0X000F) (OS) Dos32GetSharedMem Post-Invocation  
00016 (0X0010) (OS) Dos32GiveSharedMem Post-Invocation  
00017 (0X0011) (OS) Dos32QueryMem Post-Invocation  
00018 (0X0012) (OS) Dos32QueryMemState Post-Invocation  
00019 (0X0013) (OS) Dos32SetMem Post-Invocation  
00020 (0X0014) (OS) DosAllocHuge Post-Invocation  
00021 (0X0015) (OS) DosAllocProtHuge Post-Invocation  
00022 (0X0016) (OS) DosAllocProtSeg Post-Invocation  
00023 (0X0017) (OS) DosAllocSeg Post-Invocation  
00024 (0X0018) (OS) DosAllocShrProtSeg Post-Invocation  
00025 (0X0019) (OS) DosAllocShrSeg Post-Invocation  
00026 (0X001A) (OS) DosBeep Post-Invocation  
00027 (0X001B) (OS) DosBufReset Post-Invocation  
00028 (0X001C) (OS) DosCallNmPipe Post-Invocation  
00029 (0X001D) (OS) DosChDir Post-Invocation  
00030 (0X001E) (OS) DosChgFilePtr Post-Invocation  
00031 (0X001F) (OS) DosCliAccess Post-Invocation  
00032 (0X0020) (OS) DosClose Post-Invocation  
00033 (0X0021) (OS) DosCloseSem Post-Invocation  
00034 (0X0022) (OS) DosConnectNmPipe Post-Invocation  
00035 (0X0023) (OS) DosCreateCSAlias Post-Invocation  
00036 (0X0024) (OS) DosCreateSem Post-Invocation  
00037 (0X0025) (OS) DosCreateThread Post-Invocation  
00038 (0X0026) (OS) DosCWait Post-Invocation  
00039 (0X0027) (OS) DosDelete Post-Invocation  
00040 (0X0028) (OS) DosDevConfig Post-Invocation  
00041 (0X0029) (OS) DosDevIoctl2 Post-Invocation  
00042 (0X002A) (OS) DosDevIoctl Post-Invocation  
00043 (0X002B) (OS) DosDisConnectMmPipe Post-Invocation  
00044 (0X002C) (OS) DosDupHandle Post-Invocation  
00045 (0X002D) (OS) DosEditName Post-Invocation  
00047 (0X002F) (OS) DosError Pre-Invocation  
00049 (0X0031) (OS) Dos32Exit Dos Post-Invocation  
00051 (0X0033) (OS) Dos32ExitList Dos Post-Invocation  
00052 (0X0034) (OS) DosFileIO Post-Invocation  
00053 (0X0035) (OS) DosFileLocks Post-Invocation  
00054 (0X0036) (OS) DosFindClose Post-Invocation  
00055 (0X0037) (OS) DosFindFirst2 Post-Invocation  
00056 (0X0038) (OS) DosFindFirst Post-Invocation  
00057 (0X0039) (OS) DosFindFromName Post-Invocation  
00058 (0X003A) (OS) DosFindNext Post-Invocation  
00059 (0X003B) (OS) DosFindNotifyClose Post-Invocation  
00060 (0X003C) (OS) DosFindNotifyFirst Post-Invocation  
00061 (0X003D) (OS) DosFindNotifyNext Post-Invocation  
00062 (0X003E) (OS) DosFlagProcess Post-Invocation  
00063 (0X003F) (OS) DosFreeModule Post-Invocation  
00064 (0X0040) (OS) DosFreeSeg Post-Invocation  
00065 (0X0041) (OS) DosFSAttach Post-Invocation  
00066 (0X0042) (OS) DosFSCtl Post-Invocation  
00067 (0X0043) (OS) DosGetDateTime Post-Invocation  
00068 (0X0044) (OS) DosGetModHandle Post-Invocation  
00069 (0X0045) (OS) DosGetModName Post-Invocation  
00070 (0X0046) (OS) DosGetPid Post-Invocation  
00071 (0X0047) (OS) DosGetProcAddr Post-Invocation  
00072 (0X0048) (OS) DosGetPrtY Post-Invocation  
00073 (0X0049) (OS) DosGetResource Post-Invocation  
00074 (0X004A) (OS) DosGetSeg Post-Invocation  
00075 (0X004B) (OS) DosGetShrSeg Post-Invocation  
00076 (0X004C) (OS) DosGetVersion Post-Invocation  
00077 (0X004D) (OS) DosGiveSeg Post-Invocation  
00078 (0X004E) (OS) DosHoldSignal Post-Invocation

00079 (0X004F) (OS) DoslCopy Post-Invocation  
00080 (0X0050) (OS) DoslExecPgm Post-Invocation  
00083 (0X0053) (OS) DoslSetCP Post-Invocation  
00084 (0X0054) (OS) DosKillProcess Post-Invocation  
00085 (0X0055) (OS) DosLoadModule Post-Invocation  
00086 (0X0056) (OS) DosMakeNmPipe Post-Invocation  
00087 ( 0X0057) (OS) DosMakePipe Post-Invocation  
00088 (0X0058) (OS) DosMkDir2 Post-Invocation  
00089 (0X0059) (OS) DosMkDir Post-Invocation  
00095 (0X005F) (OS) DosMove Post-Invocation  
00096 (0X0060) (OS) DosMuxSemWait Post-Invocation (Waited)  
00097 (0X0061) (OS) DosNewSize Post-Invocation  
00098 (0X0062) (OS) DosOpen2 Post-Invocation  
00099 (0X0063) (OS) DosOpen Post-Invocation  
00100 (0X0064) (OS) DosOpenSem Post-Invocation  
00101 (0X0065) (OS) DosOpLockRelease Post-Invocation  
00102 (0X0066) (OS) DosOpLockWait Post-Invocation  
00103 (0X0067) (OS) DosPeekNmPipe Post-Invocation  
00104 (0X0068) (OS) DosPhysicalDisk Post-Invocation  
00105 (0X0069) (OS) DosQNmPipeState Post-Invocation  
00106 (0X006A) (OS) DosPortAccess Post-Invocation  
00108 (0X006C) (OS) DosQCurDir Post-Invocation  
00109 (0X006D) (OS) DosQCurDisk Post-Invocation  
00110 (0X006E) (OS) DosQFHandState Post-Invocation  
00111 (0X006F) (OS) DosQFileInfo Post-Invocation  
00112 (0X0070) (OS) DosQFileMode Post-Invocation  
00113 (0X0071) (OS) DosQFSAttach Post-Invocation  
00114 (0X0072) (OS) DosQFSInfo Post-Invocation  
00115 (0X0073) (OS) DosQHandType Post-Invocation  
00116 (0X0074) (OS) DosQNmPHandState Post-Invocation  
00117 (0X0075) (OS) DosQNmPipeInfo Post-Invocation  
00118 (0X0076) (OS) DosQPathInfo Post-Invocation  
00119 (0X0077) (OS) DosQSysInfo Post-Invocation  
00120 (0X0078) (OS) DosQVerify Post-Invocation  
00121 (0X0079) (OS) DosRawReadNmPipe Post-Invocation  
00122 (0X007A) (OS) DosRawWriteNmPipe Post-Invocation  
00123 (0X007B) (OS) DoslRead Post-Invocation  
00125 (0X007D) (OS) DosReallocHuge Post-Invocation  
00126 (0X007E) (OS) DosReallocSeg Post-Invocation  
00127 (0X007F) (OS) Dos32ResumeThread Dos Post-Invocation  
00128 (0X0080) (OS) DosRmdir Post-Invocation  
00129 (0X0081) (OS) DosSelectDisk Post-Invocation  
00133 (0X0085) (OS) DosSemSetWait Post-Invocation  
00135 (0X0087) (OS) DosSendSignal Post-Invocation  
00137 (0X0089) (OS) DosSetDateTime Post-Invocation  
00138 (0X008A) (OS) DosSetFHandState Post-Invocation  
00139 (0X008B) (OS) DosSetFileInfo Post-Invocation  
00140 (0X008C) (OS) DosSetFileMode Post-Invocation  
00141 (0X008D) (OS) DosSetFSInfo Post-Invocation  
00142 (0X008E) (OS) DosSetMaxFH Post-Invocation  
00143 (0X008F) (OS) DosSetNmPHandState Post-Invocation  
00144 (0X0090) (OS) DosSetNmPipeSem Post-Invocation  
00145 (0X0091) (OS) DosSetPathInfo Post-Invocation  
00146 (0X0092) (OS) DosSetPrtY Post-Invocation  
00147 (0X0093) (OS) DosSetSigHandler Post-Invocation  
00148 (0X0094) (OS) DosSetVec Post-Invocation  
00149 (0X0095) (OS) DosSetVerify Post-Invocation  
00150 (0X0096) (OS) DosSleep Post-Invocation  
00151 (0X0097) (OS) Dos32SuspendThread Dos Post-Invocation  
00152 (0X0098) (OS) DosSystemService Post-Invocation  
00153 (0X0099) (OS) DosTimerAsync Post-Invocation  
00154 (0X009A) (OS) DosTimerStart Post-Invocation  
00155 (0X009B) (OS) Dos32StopTimer Post-Invocation  
00156 (0X009C) (OS) DosTransactNmPipe Post-Invocation  
00157 (0X009D) (OS) DosWaitNmPipe Post-Invocation  
00158 (0X009E) (OS) DoslWrite Post-Invocation  
00163 (0X00A3) (OS) Dos32AliasMem Pre-Invocation  
00164 (0X00A4) (OS) Dos32AllocMem Pre-Invocation  
00165 (0X00A5) (OS) Dos32AllocProtectedMem Pre-Invocation  
00166 (0X00A6) (OS) Dos32AllocSharedMem Pre-Invocation  
00167 (0X00A7) (OS) Dos32CreateThread Pre-Invocation  
00168 (0X00A8) (OS) Dos32Debug Dos Pre-Invocation  
00169 (0X00A9) (OS) Dos32ExitList Dos Pre-Invocation

00170 (0X00AA) (OS) Dos32FreeMem Pre-Invocation  
00171 (0X00AB) (OS) Dos32GetNamedSharedMem Pre-Invocation  
00172 (0X00AC) (OS) Dos32GetSharedMem Pre-Invocation  
00173 (0X00AD) (OS) Dos32GiveSharedMem Pre-Invocation  
00174 (0X00AE) (OS) Dos32QueryMem Pre-Invocation  
00175 (0X00AF) (OS) Dos32QueryMemState Pre-Invocation  
00176 (0X00B0) (OS) Dos32SetMem Pre-Invocation  
00177 (0X00B1) (OS) DosAllocHuge Pre-Invocation  
00178 (0X00B2) (OS) DosAllocProtHuge Pre-Invocation  
00179 (0X00B3) (OS) DosAllocProtSeg Pre-Invocation  
00180 (0X00B4) (OS) DosAllocSeg Pre-Invocation  
00181 (0X00B5) (OS) DosAllocShrProtSeg Pre-Invocation  
00182 (0X00B6) (OS) DosAllocShrSeg Pre-Invocation  
00183 (0X00B7) (OS) DosBeep Pre-Invocation  
00184 (0X00B8) (OS) DosBufReset Pre-Invocation  
00185 (0X00B9) (OS) DosCallNmPipe Pre-Invocation  
00186 (0X00BA) (OS) DosChDir Pre-Invocation  
00187 (0X00BB) (OS) DosChgFilePtr Pre-Invocation  
00188 (0X00BC) (OS) DosCliAccess Pre-Invocation  
00189 (0X00BD) (OS) DosClose Pre-Invocation  
00190 (0X00BE) (OS) DosCloseSem Pre-Invocation  
00191 (0X00BF) (OS) DosConnectNmPipe Pre-Invocation  
00192 (0X00C0) (OS) DosCreateCSAlias Pre-Invocation  
00193 (0X00C1) (OS) DosCreateSem Pre-Invocation  
00194 (0X00C2) (OS) DosCreateThread Pre-Invocation  
00195 (0X00C3) (OS) DosCWait Pre-Invocation  
00196 (0X00C4) (OS) DosDelete Pre-Invocation  
00197 (0X00C5) (OS) DosDevConfig Pre-Invocation  
00198 (0X00C6) (OS) DosDevIoctl2 Pre-Invocation  
00199 (0X00C7) (OS) DosDevIoctl Pre-Invocation  
00200 (0X00C8) (OS) DosDisConnectMmPipe Pre-Invocation  
00201 (0X00C9) (OS) DosDupHandle Pre-Invocation  
00202 (0X00CA) (OS) DosEditName Pre-Invocation  
00203 (0X00CB) (OS) DosEnterCritSec Pre-Invocation  
00204 (0X00CC) (OS) DosError Pre-Invocation  
00206 (0X00CE) (OS) DosExit Dos Pre-Invocation  
00207 (0X00CF) (OS) DosExitCritSec Pre-Invocation  
00208 (0X00D0) (OS) DosExitList Dos Pre-Invocation  
00209 (0X00D1) (OS) DosFileIO Pre-Invocation  
00210 (0X00D2) (OS) DosFileLocks Pre-Invocation  
00211 (0X00D3) (OS) DosFindClose Pre-Invocation  
00212 (0X00D4) (OS) DosFindFirst2 Pre-Invocation  
00213 (0X00D5) (OS) DosFindFirst Pre-Invocation  
00214 (0X00D6) (OS) DosFindFromName Pre-Invocation  
00215 (0X00D7) (OS) DosFindNext Pre-Invocation  
00216 (0X00D8) (OS) DosFindNotifyClose Pre-Invocation  
00217 (0X00D9) (OS) DosFindNotifyFirst Pre-Invocation  
00218 (0X00DA) (OS) DosFindNotifyNext Pre-Invocation  
00219 (0X00DB) (OS) DosFlagProcess Pre-Invocation  
00220 (0X00DC) (OS) DosFreeModule Pre-Invocation  
00221 (0X00DD) (OS) DosFreeSeg Pre-Invocation  
00222 (0X00DE) (OS) DosFSAttach Pre-Invocation  
00223 (0X00DF) (OS) DosFSCtl Pre-Invocation  
00224 (0X00E0) (OS) DosGetDateTime Pre-Invocation  
00225 (0X00E1) (OS) DosGetModHandle Pre-Invocation  
00226 (0X00E2) (OS) DosGetModName Pre-Invocation  
00227 (0X00E3) (OS) DosGetPid Pre-Invocation  
00228 (0X00E4) (OS) DosGetProcAddr Pre-Invocation  
00229 (0X00E5) (OS) DosGetPrtY Pre-Invocation  
00230 (0X00E6) (OS) DosGetResource Pre-Invocation  
00231 (0X00E7) (OS) DosGetSeg Pre-Invocation  
00232 (0X00E8) (OS) DosGetShrSeg Pre-Invocation  
00233 (0X00E9) (OS) DosGetVersion Pre-Invocation  
00234 (0X00EA) (OS) DosGiveSeg Pre-Invocation  
00235 (0X00EB) (OS) DosHoldSignal Pre-Invocation  
00236 (0X00EC) (OS) DosICopy Pre-Invocation  
00237 (0X00ED) (OS) DosIExecPgm Pre-Invocation  
00240 (0X00F0) (OS) DosISetCP Pre-Invocation  
00241 (0X00F1) (OS) DosKillProcess Pre-Invocation  
00242 (0X00F2) (OS) DosLoadModule Pre-Invocation  
00243 (0X00F3) (OS) DosMakeNmPipe Pre-Invocation  
00244 (0X00F4) (OS) DosMakePipe Pre-Invocation  
00245 (0X00F5) (OS) DosMkDir2 Pre-Invocation

00246 (0X00F6) (OS) DosMkDir Pre-Invocation  
00252 (0X00FC) (OS) DosMove Pre-Invocation  
00253 (0X00FD) (OS) DosMuxSemWait Pre-Invocation  
00254 (0X00FE) (OS) DosNewSize Pre-Invocation  
00255 (0X00FF) (OS) DosOpen2 Pre-Invocation  
00256 (0X0100) (OS) DosOpen Pre-Invocation  
00257 (0X0101) (OS) DosOpenSem Pre-Invocation  
00258 (0X0102) (OS) DosOpLockRelease Pre-Invocation  
00259 (0X0103) (OS) DosOpLockWait Pre-Invocation  
00260 (0X0104) (OS) DosPeekNmPipe Pre-Invocation  
00261 (0X0105) (OS) DosPhysicalDisk Pre-Invocation  
00262 (0X0106) (OS) DosQNmPipeState Pre-Invocation  
00263 (0X0107) (OS) DosPortAccess Pre-Invocation  
00265 (0X0109) (OS) DosQCurDir Pre-Invocation  
00266 (0X010A) (OS) DosQCurDisk Pre-Invocation  
00267 (0X010B) (OS) DosQFHandState Pre-Invocation  
00268 (0X010C) (OS) DosQFileInfo Pre-Invocation  
00269 (0X010D) (OS) DosQFileMode Pre-Invocation  
00270 ( 0X010E) (OS) DosQFSAttach Pre-Invocation  
00271 (0X010F) (OS) DosQFSInfo Pre-Invocation  
00272 (0X0110) (OS) DosQHandType Pre-Invocation  
00273 (0X0111) (OS) DosQNmPHandState Pre-Invocation  
00274 (0X0112) (OS) DosQNmPipeInfo Pre-Invocation  
00275 (0X0113) (OS) DosQPathInfo Pre-Invocation  
00276 (0X0114) (OS) DosQSysInfo Pre-Invocation  
00277 (0X0115) (OS) DosQVerify Pre-Invocation  
00278 (0X0116) (OS) DosRawReadNmPipe Pre-Invocation  
00279 (0X0117) (OS) DosRawWriteNmPipe Pre-Invocation  
00280 (0X0118) (OS) DosIRead Pre-Invocation  
00282 (0X011A) (OS) DosReallocHuge Pre-Invocation  
00283 (0X011B) (OS) DosReallocSeg Pre-Invocation  
00284 (0X011C) (OS) DosResumeThread Dos Pre-Invocation  
00285 (0X011D) (OS) DosRmdir Pre-Invocation  
00286 (0X011E) (OS) DosSelectDisk Pre-Invocation  
00290 (0X0122) (OS) DosSemSetWait Pre-Invocation  
00292 (0X0124) (OS) DosSendSignal Pre-Invocation  
00294 (0X0126) (OS) DosSetDateTime Pre-Invocation  
00295 (0X0127) (OS) DosSetFHandState Pre-Invocation  
00296 (0X0128) (OS) DosSetFileInfo Pre-Invocation  
00297 (0X0129) (OS) DosSetFileMode Pre-Invocation  
00298 (0X012A) (OS) DosSetFSInfo Pre-Invocation  
00299 (0X012B) (OS) DosSetMaxFH Pre-Invocation  
00300 (0X012C) (OS) DosSetNmPHandState Pre-Invocation  
00301 (0X012D) (OS) DosSetNmPipeSem Pre-Invocation  
00302 (0X012E) (OS) DosSetPathInfo Pre-Invocation  
00303 (0X012F) (OS) DosSetPrt Pre-Invocation  
00304 (0X0130) (OS) DosSetSigHandler Pre-Invocation  
00305 (0X0131) (OS) DosSetVec Pre-Invocation  
00306 (0X0132) (OS) DosSetVerify Pre-Invocation  
00307 (0X0133) (OS) DosSleep Pre-Invocation  
00308 (0X0134) (OS) DosSuspendThread Dos Pre-Invocation  
00309 (0X0135) (OS) DosSystemService Pre-Invocation  
00310 (0X0136) (OS) DosTimerAsync Pre-Invocation  
00311 (0X0137) (OS) DosTimerStart Pre-Invocation  
00312 (0X0138) (OS) Dos32StopTimer Pre-Invocation  
00313 (0X0139) (OS) DosTransactNmPipe Pre-Invocation  
00314 (0X013A) (OS) DosWaitNmPipe Pre-Invocation  
00315 (0X013B) (OS) DosIWrite Pre-Invocation  
00323 (0X0143) (OS) DosGetResource2 Pre-Invocation  
00324 (0X0144) (OS) DosGetResource2 Post-Invocation  
00325 (0X0145) (OS) DosFreeResource Pre-Invocation  
00326 (0X0146) (OS) DosFreeResource Post-Invocation  
00327 (0X0147) (OS) Dos32GetResource Pre-Invocation  
00328 (0X0148) (OS) Dos32GetResource Post-Invocation  
00331 (0X014B) (OS) Dos32FreeRsource Pre-Invocation  
00332 (0X014C) (OS) Dos32FreeResource Post-Invocation  
00333 (0X014D) (OS) Dos32QueryProcAddr Pre-Invocation  
00334 (0X014E) (OS) Dos32QueryProcAddr Post-Invocation  
00335 (0X014F) (OS) Dos32CreateEventSem Pre-Invocation  
00336 (0X0150) (OS) Post-Invocation  
00337 (0X0151) (OS) Dos32OpenEventSem Pre-Invocation  
00338 (0X0152) (OS) Post-Invocation  
00339 (0X0153) (OS) Dos32OpenEventSem Pre-Invocation



00340 (0X0154) (OS) Dos32CloseEventSem Post-Invocation  
00341 (0X0155) (OS) Dos32ResetEventSem Pre-Invocation  
00342 (0X0156) (OS) Dos32ResetEventSem Post-Invocation  
00343 (0X0157) (OS) Dos32PostEventSem Pre-Invocation  
00344 (0X0158) (OS) Dos32PostEventSem Post-Invocation  
00345 (0X0159) (OS) Dos32WaitEvenSem Pre-Invocation  
00346 (0X015A) (OS) Dos32WaitEventSem Post-Invocation  
00347 (0X015B) (OS) Dos32QueryEventSem Pre-Invocation  
00348 (0X015C) (OS) Dos32QueryEventSem Post-Invocation  
00349 (0X015D) (OS) Dos32CreateMutexSem Pre-Invocation  
00350 (0X015E) (OS) Dos32CreateMutexSem Post-Invocation  
00351 (0X015F) (OS) Dos32OpenMutexSem Pre-Invocation  
00352 (0X0160) (OS) Dos32OpenMutexSem Post-Invocation  
00353 (0X0161) (OS) Dos32CloseMutexSem Pre-Invocation  
00354 (0X0162) (OS) Dos32CloseMutexSem Post-Invocation  
00355 (0X0163) (OS) Dos32RequestMutexSem Pre-Invocation  
00356 (0X0164) (OS) Dos32RequestMutexSem Post-Invocation  
00357 (0X0165) (OS) Dos32ReleaseMutexSem Pre-Invocation  
00358 (0X0166) (OS) Dos32ReleaseMutexSem Post-Invocation  
00359 (0X0167) (OS) Dos32QueryMutexSem Pre-Invocation  
00360 (0X0168) (OS) Dos32QueryMutexSem Post-Invocation  
00361 (0X0169) (OS) Dos32CreateMuxWaitSem Pre-Invocation  
00362 (0X016A) (OS) Dos32CreateMuxWaitSem Post-Invocation  
00363 (0X016B) (OS) Dos32OpenMuxWaitSem Pre-Invocation  
00364 (0X016C) (OS) Dos32OpenMuxWaitSem Post-Invocation  
00365 (0X016D) (OS) Dos32CloseMuxWaitSem Pre-Invocation  
00366 (0X016E) (OS) Dos32CloseMuxWaitSem Post-Invocation  
00367 (0X016F) (OS) Dos32WaitMuxWaitSem Pre-Invocation  
00368 (0X0170) (OS) Dos32WaitMuxWaitSem Post-Invocation  
00369 (0X0171) (OS) Dos32AddMuxWaitSem Pre-Invocation  
00370 (0X0172) (OS) Dos32AddMuxWaitSem Post-Invocation  
00371 (0X0173) (OS) Dos32DeleteMuxWaitSem Pre-Invocation  
00372 (0X0174) (OS) Dos32DeleteMuxWaitSem Post-Invocation  
00373 (0X0175) (OS) Dos32QueryMuxWaitSem Pre-Invocation  
00374 (0X0176) (OS) Dos32QueryMuxWaitSem Post-Invocation  
00375 (0X0177) (OS) DosGetCP Pre-Invocation  
00376 (0X0178) (OS) DosGetCP Post-Invocation  
00377 (0X0179) (OS) Dos32AsyncTimer Pre-Invocation  
00378 (0X017A) (OS) Dos32AsyncTimer Post-Invocation  
00379 (0X017B) (OS) Dos32StartTimer Pre-Invocation  
00380 (0X017C) (OS) Dos32StartTimer Post-Invocation  
00391 (0X0187) (OS) Dos32WaitThread Pre-Invocation  
00392 (0X0188) (OS) Dos32WaitThread Post-Invocation  
00393 (0X0189) (OS) Cluster Allocate Pre-Invocation  
00394 (0X018A) (OS) Cluster Allocate Post-Invocation  
00395 (0X018B) (OS) Cluster Release Pre-Invocation  
00397 (0X018D) (OS) File Lock/Unlock Pre-Invocation  
00401 (0X0191) (OS) Thread Reschedule Post-Invocation  
00403 (0X0193) (OS) DosMuxSemWait Post-Invocation (No Wait)  
00404 (0X0194) (OS) DosEnumAttribute Pre-Invocation  
00405 (0X0195) (OS) DosEnumAttribute Post-Invocation  
00406 (0X0196) (OS) DoslSetFileInfo Pre-Invocation  
00407 (0X0197) (OS) DoslSetFileInfo Post-Invocation  
00408 (0X0198) (OS) DoslSetPathInfo Pre-Invocation  
00409 (0X0199) (OS) DoslSetPathInfo Post-Invocation  
00410 (0X019A) (OS) Dos32QueryResource Pre-Invocation  
00411 (0X019B) (OS) Dos32QueryResourceSize Post-Invocation  
00412 (0X019C) (OS) DoslDevloctl Pre-Invocation  
00413 (0X019D) (OS) DoslDevloctl Post-Invocation  
00414 (0X019E) (OS) DoslSetRelMaxFH Pre-Invocation  
00415 (0X019F) (OS) DoslSetRelMaxFH Post-Invocation  
00416 (0X01A0) (OS) Dos32InitializePorthole Pre-Invocation  
00417 (0X01A1) (OS) Dos32InitializePorthole Post-Invocation  
00418 (0X01A2) (OS) Dos32QueryHeaderInfo Pre-Invocation  
00419 (0X01A3) (OS) Dos32QueryHeaderInfo Post-Invocation  
00420 (0X01A4) (OS) Dos32QueryProcType Pre-Invocation  
00421 (0X01A5) (OS) Dos32QueryProcType Post-Invocation  
00424 (0X01A8) (OS) DosOpen2Compt Pre-Invocation  
00425 (0X01A9) (OS) DosOpen2Compt Post-Invocation  
00428 (0X01AC) (OS) Dos32lSetFHState Pre-Invocation  
00429 (0X01AD) (OS) Dos32lSetFHState Post-Invocation  
00430 (0X01AE) (OS) Dos32lQUERYFHSTATE Pre-Invocation  
00431 (0X01AF) (OS) Dos32lQUERYFHSTATE Post-Invocation

00432 (0X01B0) (OS) Dos32IRead Pre-Invocation  
00433 (0X01B1) (OS) Dos32IRead Post-Invocation  
00434 (0X01B2) (OS) Dos32IWrite Pre-Invocation  
00435 (0X01B3) (OS) Dos32IWrite Post-Invocation  
00436 (0X01B4) (OS) Dos32DumpProcess Pre-Invocation  
00437 (0X01B5) (OS) Dos32DumpProcess Post-Invocation  
00438 (0X01B6) (OS) Dos32SuppressPopUps Pre-Invocation  
00439 (0X01B7) (OS) Dos32SuppressPopUps Post-Invocation  
00440 (0X01B8) (OS) Dos32KillThread Pre-Invocation  
00441 (0X01B9) (OS) Dos32KillThread Post-Invocation  
00442 (0X01BA) (OS) Dos32IProtectSetFHState Pre-Invocation  
00443 (0X01BB) (OS) Dos32IProtectSetFHState Post-Invocation  
00444 (0X01BC) (OS) Dos32IPROTECTQUERYFHSTATE Pre-Invocation  
00445 (0X01BD) (OS) Dos32IPROTECTQUERYFHSTATE Post-Invocation  
00446 (0X01BE) (OS) DosProtectChgFilePtr Pre-Invocation  
00447 (0X01BF) (OS) DosProtectChgFilePtr Post-Invocation  
00448 (0X01C0) (OS) DosProtectClose Pre-Invocation  
00449 (0X01C1) (OS) DosProtectClose Post-Invocation  
00450 (0X01C2) (OS) DosCloseChangeNotify Pre-Invocation  
00451 (0X01C3) (OS) DosCloseChangeNotify Post-Invocation  
00452 (0X01C4) (OS) DosProtectEnumAttribute Pre-Invocation  
00453 (0X01C5) (OS) DosProtectEnumAttribute Post-Invocation  
00454 (0X01C6) (OS) DosProtectFileIO Pre-Invocation  
00455 (0X01C7) (OS) DosProtectFileIO Post-Invocation  
00456 (0X01C8) (OS) DosProtectFileLocks Pre-Invocation  
00457 (0X01C9) (OS) DosProtectFileLocks Post-Invocation  
00458 (0X01CA) (OS) DosForceDelete Pre-Invocation  
00459 (0X01CB) (OS) DosForceDelete Post-Invocation  
00460 (0X01CC) (OS) DosIProtectRead Pre-Invocation  
00461 (0X01CD) (OS) DosIProtectRead Post-Invocation  
00462 (0X01CE) (OS) DosIProtectSetFileInfo Pre-Invocation  
00463 (0X01CF) (OS) DosIProtectSetFileInfo Post-Invocation  
00464 (0X01D0) (OS) DosIProtectWrite Pre-Invocation  
00465 (0X01D1) (OS) DosIProtectWrite Post-Invocation  
00466 (0X01D2) (OS) DosProtectNewSize Pre-Invocation  
00467 (0X01D3) (OS) DosProtectNewSize Post-Invocation  
00468 (0X01D4) (OS) DOSPROTECTOPEN Pre-Invocation  
00469 (0X01D5) (OS) DosProtectOpen Post-Invocation  
00470 (0X01D6) (OS) DosOpenChangeNotify Pre-Invocation  
00471 (0X01D7) (OS) DosOpenChangeNotify Post-Invocation  
00472 (0X01D8) (OS) DosProtectQFHandState Pre-Invocation  
00473 (0X01D9) (OS) DosProtectQFHandState Post-Invocation  
00474 (0X01DA) (OS) DosProtectQFileInfo Pre-Invocation  
00475 (0X01DB) (OS) DosProtectQFileInfo Post-Invocation  
00476 (0X01DC) (OS) DosResetChangeNotify Pre-Invocation  
00477 (0X01DD) (OS) DosResetChangeNotify Post-Invocation  
00478 (0X01DE) (OS) DosProtectSetFHandState Pre-Invocation  
00479 (0X01DF) (OS) DosProtectSetFHandState Post-Invocation  
00480 (0X01E0) (OS) DosProtectSetFileInfo Pre-Invocation  
00481 (0X01E1) (OS) DosProtectSetFileInfo Post-Invocation  
00482 (0X01E2) (OS) Dos32PMPostEventSem Pre-Invocation  
00483 (0X01E3) (OS) Dos32PMPostEventSem Post-Invocation  
00484 (0X01E4) (OS) Dos32PMWaitEventSem Post-Invocation  
00485 (0X01E5) (OS) Dos32PMWaitMuxWaitSem Pre-Invocation  
00486 (0X01E6) (OS) Dos32PMWaitMuxWaitSem Post-Invocation  
00487 (0X01E7) (OS) Dos32QueryExtLIBPATH Pre-Invocation  
00488 (0X01E8) (OS) Dos32QueryExtLIBPATH Post-Invocation  
00489 (0X01E9) (OS) Dos32SetExtLIBPATH Pre-Invocation  
00490 (0X01EA) (OS) Dos32SetExtLIBPATH Post-Invocation  
00491 (0X01EB) (OS) Dos32VERIFYPIDTID Pre-Invocation  
00492 (0X01EC) (OS) Dos32VERIFYPIDTID Post-Invocation  
00493 (0X01ED) (OS) Dos32PMWaitEventSem Pre-Invocation  
00494 (0X01EE) (OS) Dos32CancelLockRequest Pre-Invocation  
00495 (0X01EF) (OS) Dos32CancelLockRequest Post-Invocation  
00496 (0X01F0) (OS) Dos32SetFileLocks Pre-Invocation  
00497 (0X01F1) (OS) Dos32SetFileLocks Post-Invocation  
00498 (0X01F2) (OS) Dos32ProtectSetFileLocks Pre-Invocation  
00499 (0X01F3) (OS) Dos32ProtectSetFileLocks Post-Invocation  
00500 (0X01F4) (OS) DosCreateSpinLock Pre-Invocation  
00501 (0X01F5) (OS) DosCreateSpinLock Post-Invocation  
00502 (0X01F6) (OS) DosAcquireSpinLock Pre-Invocation  
00503 (0X01F7) (OS) DosAcquireSpinLock Post-Invocation  
00504 (0X01F8) (OS) DosReleaseSpinLock Pre-Invocation

00505 (0X01F9) (OS) DosReleaseSpinLock Post-Invocation  
00506 (0X01FA) (OS) DosFreeSpinLock Pre-Invocation  
00507 (0X01FB) (OS) DosFreeSpinLock Post-Invocation  
00508 (0X01FC) (OS) Dos32IProtectRead Pre-Invocation  
00509 (0X01FD) (OS) Dos32IProtectRead Post-Invocation  
00510 (0X01FE) (OS) Dos32IProtectWrite Pre-Invocation  
00511 (0X01FF) (OS) Dos32IProtectWrite Post-Invocation  
32768 (0X8000) (OS) DosSetTraceInfo Pre-Invocation  
65521 (0XFFF1) (OS) DosSetTraceInfo Post-Invocation

---

## Trace Events for OS2KRNL Major Code: 0X0005, Sorted by Tracepoint

Allocate 00393 (0X0189)  
DOSMUXSEMWAIT 00253 (0X00FD)  
post2DOSENTERCRITSEC 00001 (0X0001)  
post2DOSEXITCRITSEC 00002 (0X0002)  
post2DOSHOLD SIGNAL 00003 (0X0003)  
postAllocate 00394 (0X018A)  
postDOS32ADDMUXWAITSEM 00370 (0X0172)  
postDOS32ALIASMEM 00006 (0X0006)  
postDOS32ALLOCMEM 00007 (0X0007)  
postDOS32ALLOCPROTECTEDMEM 00008 (0X0008)  
postDOS32ALLOCSHAREDMEM 00009 (0X0009)  
postDOS32ASYNCTIMER 00378 (0X017A)  
postDOS32CANCELLOCKREQUEST 00495 (0X01EF)  
postDOS32CLOSEEVENTSEM 00340 (0X0154)  
postDOS32CLOSEMUTEXSEM 00354 (0X0162)  
postDOS32CLOSEMUXWAITSEM 00366 (0X016E)  
postDOS32CREATEEVENTSEM 00336 (0X0150)  
postDOS32CREATEMUTEXSEM 00350 (0X015E)  
postDOS32CREATEMUXWAITSEM 00362 (0X016A)  
postDOS32CREATETHREAD 00010 (0X000A)  
postDOS32DEBUG 00011 (0X000B)  
postDOS32DELETEMUXWAITSEM 00372 (0X0174)  
postDOS32DUMPPROCESS 00437 (0X01B5)  
postDOS32EXITLIST 00012 (0X000C)  
postDOS32FREEMEM 00013 (0X000D)  
postDOS32FREERESOURCE 00332 (0X014C)  
postDOS32GETNAMEDSHAREDMEM 00014 (0X000E)  
postDOS32GETRESOURCE 00328 (0X0148)  
postDOS32GETSHAREDMEM 00015 (0X000F)  
postDOS32GIVESHAREDMEM 00016 (0X0010)  
postDOS32IASYNCTIMER 00153 (0X0099)  
postDOS32INITIALIZEPORTHOLE 00417 (0X01A1)  
postDOS32IPROTECTQUERYFHSTATE 00445 (0X01BD)  
postDOS32IPROTECTREAD 00509 (0X01FD)  
postDOS32IPROTECTSETFHSTATE 00443 (0X01BB)  
postDOS32IPROTECTWRITE 00511 (0X01FF)  
postDOS32IQUERYFHSTATE 00431 (0X01AF)  
postDOS32IREAD 00433 (0X01B1)  
postDOS32ISETFHSTATE 00429 (0X01AD)  
postDOS32ISTARTTIMER 00154 (0X009A)  
postDOS32IWRITE 00435 (0X01B3)  
postDOS32KILLTHREAD 00441 (0X01B9)  
postDOS32OPENEVENTSEM 00338 (0X0152)  
postDOS32OPENMUTEXSEM 00352 (0X0160)  
postDOS32OPENMUXWAITSEM 00364 (0X016C)  
postDOS32PMPOSTEVENTSEM 00483 (0X01E3)  
postDOS32PMWAITEVENTSEM 00484 (0X01E4)  
postDOS32PMWAITMUXWAITSEM 00486 (0X01E6)  
postDOS32POSTEVENTSEM 00344 (0X0158)  
postDOS32PROTECTSETFILELOCKS 00499 (0X01F3)  
postDOS32QUERYEVENTSEM 00348 (0X015C)

postDOS32QUERYEXTLIBPATH 00488 (0X01E8)  
postDOS32QUERYHEADERINFO 00419 (0X01A3)  
postDOS32QUERYMEM 00017 (0X0011)  
postDOS32QUERYMEMSTATE 00018 (0X0012)  
postDOS32QUERYMUTEXSEM 00360 (0X0168)  
postDOS32QUERYMUXWAITSEM 00374 (0X0176)  
postDOS32QUERYPROCADDR 00334 (0X014E)  
postDOS32QUERYPROCTYPE 00421 (0X01A5)  
postDOS32QUERYRESOURCESIZE 00411 (0X019B)  
postDOS32RELEASEMUTEXSEM 00358 (0X0166)  
postDOS32REQUESTMUTEXSEM 00356 (0X0164)  
postDOS32RESETEVENTSEM 00342 (0X0156)  
postDOS32SETEXTLIBPATH 00490 (0X01EA)  
postDOS32SETFILELOCKS 00497 (0X01F1)  
postDOS32SETMEM 00019 (0X0013)  
postDOS32STARTTIMER 00380 (0X017C)  
postDOS32STOPTIMER 00155 (0X009B)  
postDOS32SUPPRESSPOPUPS 00439 (0X01B7)  
postDOS32VERIFYPIDTID 00492 (0X01EC)  
postDOS32WAITEVENTSEM 00346 (0X015A)  
postDOS32WAITMUXWAITSEM 00368 (0X0170)  
postDOS32WAITTHREAD 00392 (0X0188)  
postDOSACQUIRESPINLOCK 00503 (0X01F7)  
postDOSALLOC HUGE 00020 (0X0014)  
postDOSALLOC PROTHUGE 00021 (0X0015)  
postDOSALLOC PROTSEG 00022 (0X0016)  
postDOSALLOC SEG 00023 (0X0017)  
postDOSALLOC SHRPROTSEG 00024 (0X0018)  
postDOSALLOC SHRSEG 00025 (0X0019)  
postDOSBEEP 00026 (0X001A)  
postDOSBUFRESET 00027 (0X001B)  
postDOSCALLNMPPIPE 00028 (0X001C)  
postDOSCHDIR 00029 (0X001D)  
postDOSCHGFILEPTR 00030 (0X001E)  
postDOSCLIACCESS 00031 (0X001F)  
postDOSCLOSE 00032 (0X0020)  
postDOSCLOSECHANGENOTIFY 00451 (0X01C3)  
postDOSCLOSESEM 00033 (0X0021)  
postDOSCONNECTNMPPIPE 00034 (0X0022)  
postDOSCREATECSALIAS 00035 (0X0023)  
postDOSCREATESEM 00036 (0X0024)  
postDOSCREATESPINLOCK 00501 (0X01F5)  
postDOSCREATETHREAD 00037 (0X0025)  
postDOSCWAIT 00038 (0X0026)  
postDOSDELETE 00039 (0X0027)  
postDOSDEVCONFIG 00040 (0X0028)  
postDOSDEVIOTL 00042 (0X002A)  
postDOSDEVIOTL2 00041 (0X0029)  
postDOSDISCONNECTNMPPIPE 00043 (0X002B)  
postDOSDUPHANDLE 00044 (0X002C)  
postDOSEDITNAME 00045 (0X002D)  
postDOSENUMATTRIBUTE 00405 (0X0195)  
postDOSERROR 00047 (0X002F)  
postDOSEXIT 00049 (0X0031)  
postDOSEXITLIST 00051 (0X0033)  
postDOSFILEIO 00052 (0X0034)  
postDOSFILELOCKS 00053 (0X0035)  
postDOSFINDCLOSE 00054 (0X0036)  
postDOSFINDFIRST 00056 (0X0038)  
postDOSFINDFIRST2 00055 (0X0037)  
postDOSFINDFROMNAME 00057 (0X0039)  
postDOSFINDNEXT 00058 (0X003A)  
postDOSFINDNOTIFYCLOSE 00059 (0X003B)  
postDOSFINDNOTIFYFIRST 00060 (0X003C)  
postDOSFINDNOTIFYNEXT 00061 (0X003D)  
postDOSFLAGPROCESS 00062 (0X003E)  
postDOSFORCEDELETE 00459 (0X01CB)  
postDOSFREEMODULE 00063 (0X003F)  
postDOSFREERESOURCE 00326 (0X0146)  
postDOSFREESEG 00064 (0X0040)  
postDOSFREESPINLOCK 00507 (0X01FB)  
postDOSFSATTACH 00065 (0X0041)  
postDOSFSCTL 00066 (0X0042)

postDOSGETCP 00376 (0X0178)  
postDOSGETDATETIME 00067 (0X0043)  
postDOSGETMODHANDLE 00068 (0X0044)  
postDOSGETMODNAME 00069 (0X0045)  
postDOSGETPID 00070 (0X0046)  
postDOSGETPROCADDR 00071 (0X0047)  
postDOSGETPRTY 00072 (0X0048)  
postDOSGETRESOURCE 00073 (0X0049)  
postDOSGETRESOURCE2 00324 (0X0144)  
postDOSGETSEG 00074 (0X004A)  
postDOSGETSHRSEG 00075 (0X004B)  
postDOSGETVERSION 00076 (0X004C)  
postDOSGIVESEG 00077 (0X004D)  
postDOSHOLD SIGNAL 00078 (0X004E)  
postDOSICOPY 00079 (0X004F)  
postDOSIDEVIOCTL 00413 (0X019D)  
postDOSIEXEC PGM 00080 (0X0050)  
postDOSIPROTECTREAD 00461 (0X01CD)  
postDOSIPROTECTSETFILEINFO 00463 (0X01CF)  
postDOSIPROTECTWRITE 00465 (0X01D1)  
postDOSIREAD 00123 (0X007B)  
postDOSISETCP 00083 (0X0053)  
postDOSISETFILEINFO 00407 (0X0197)  
postDOSISETPATHINFO 00409 (0X0199)  
postDOSISETRELMAXFH 00415 (0X019F)  
postDOSIWRITE 00158 (0X009E)  
postDOSKILLPROCESS 00084 (0X0054)  
postDOSLOADMODULE 00085 (0X0055)  
postDOSMAKENMPIPE 00086 (0X0056)  
postDOSMAKEPIPE 00087 (0X0057)  
postDOSMKDIR 00089 (0X0059)  
postDOSMKDIR2 00088 (0X0058)  
postDOSMOVE 00095 (0X005F)  
postDOSMUXSEMWAIT 00096 (0X0060)  
postDOSMUXSEMWAIT2 00403 (0X0193)  
postDOSNEWSIZE 00097 (0X0061)  
postDOSOPEN 00099 (0X0063)  
postDOSOPEN2 00098 (0X0062)  
postDOSOPEN2COMPT 00425 (0X01A9)  
postDOSOPENCHANGENOTIFY 00471 (0X01D7)  
postDOSOPENSEM 00100 (0X0064)  
postDOSOPLOCKRELEASE 00101 (0X0065)  
postDOSOPLOCKWAIT 00102 (0X0066)  
postDOSPEEKNMPIPE 00103 (0X0067)  
postDOSPHYSICALDISK 00104 (0X0068)  
postDOSPORTACCESS 00106 (0X006A)  
postDOSPROTECTCHGFILEPTR 00447 (0X01BF)  
postDOSPROTECTCLOSE 00449 (0X01C1)  
postDOSPROTECTENUMATTRIBUTE 00453 (0X01C5)  
postDOSPROTECTFILEIO 00455 (0X01C7)  
postDOSPROTECTFILELOCKS 00457 (0X01C9)  
postDOSPROTECTNEWSIZE 00467 (0X01D3)  
postDOSPROTECTOPEN 00469 (0X01D5)  
postDOSPROTECTQFHANDSTATE 00473 (0X01D9)  
postDOSPROTECTQFILEINFO 00475 (0X01DB)  
postDOSPROTECTSETFHANDSTATE 00479 (0X01DF)  
postDOSPROTECTSETFILEINFO 00481 (0X01E1)  
postDOSQCURDIR 00108 (0X006C)  
postDOSQCURDISK 00109 (0X006D)  
postDOSQFHANDSTATE 00110 (0X006E)  
postDOSQFILEINFO 00111 (0X006F)  
postDOSQFILEMODE 00112 (0X0070)  
postDOSQFSATTACH 00113 (0X0071)  
postDOSQFSINFO 00114 (0X0072)  
postDOSQHANDTYPE 00115 (0X0073)  
postDOSQNMPIPEHANDSTATE 00116 (0X0074)  
postDOSQNMPIPEINFO 00117 (0X0075)  
postDOSQNMPIPESEMSTATE 00105 (0X0069)  
postDOSQPATHINFO 00118 (0X0076)  
postDOSQSYSINFO 00119 (0X0077)  
postDOSQVERIFY 00120 (0X0078)  
postDOSRAWREADNMPIPE 00121 (0X0079)  
postDOSRAWWRITENMPIPE 00122 (0X007A)

postDOSREALLOCHUGE 00125 (0X007D)  
postDOSREALLOCSEG 00126 (0X007E)  
postDOSRELEASESPINLOCK 00505 (0X01F9)  
postDOSRESETCHANGENOTIFY 00477 (0X01DD)  
postDOSRESUMETHREAD 00127 (0X007F)  
postDOSRMDIR 00128 (0X0080)  
postDOSSELECTDISK 00129 (0X0081)  
postDOSSEMSETWAIT 00133 (0X0085)  
postDOSSEENDSIGNAL 00135 (0X0087)  
postDOSSETDATETIME 00137 (0X0089)  
postDOSSETFHANDSTATE 00138 (0X008A)  
postDOSSETFILEINFO 00139 (0X008B)  
postDOSSETFILEMODE 00140 (0X008C)  
postDOSSETFSINFO 00141 (0X008D)  
postDOSSETMAXFH 00142 (0X008E)  
postDOSSETNMPIHANDSTATE 00143 (0X008F)  
postDOSSETNMPIPESEM 00144 (0X0090)  
postDOSSETPATHINFO 00145 (0X0091)  
postDOSSETPRTY 00146 (0X0092)  
postDOSSETSIGHANDLER 00147 (0X0093)  
postDOSSETTRACEINFO 65521 (0XFFF1)  
postDOSSETVEC 00148 (0X0094)  
postDOSSETVERIFY 00149 (0X0095)  
postDOSSLEEP 00150 (0X0096)  
postDOSSUSPENDTHREAD 00151 (0X0097)  
postDOSSYSTEMSERVICE 00152 (0X0098)  
postDOSTRANSACTNMPIPE 00156 (0X009C)  
postDOSWAITNMPIPE 00157 (0X009D)  
postSchedNext 00401 (0X0191)  
preDOS32ADDMUXWAITSEM 00369 (0X0171)  
preDOS32ALIASMEM 00163 (0X00A3)  
preDOS32ALLOCMEM 00164 (0X00A4)  
preDOS32ALLOCPROTECTEDMEM 00165 (0X00A5)  
preDOS32ALLOCSHAREDMEM 00166 (0X00A6)  
preDOS32ASYNCTIMER 00377 (0X0179)  
preDOS32CLOSEEVENTSEM 00339 (0X0153)  
preDOS32CLOSEMUTEXSEM 00353 (0X0161)  
preDOS32CLOSEMUXWAITSEM 00365 (0X016D)  
preEDOS32CANCELLOCKREQUEST 00494 (0X01EE)  
preDOS32CREATEEVENTSEM 00335 (0X014F)  
preDOS32CREATEMUTEXSEM 00349 (0X015D)  
preDOS32CREATEMUXWAITSEM 00361 (0X0169)  
preDOS32CREATETHREAD 00167 (0X00A7)  
preDOS32DEBUG 00168 (0X00A8)  
preDOS32DELETEMUXWAITSEM 00371 (0X0173)  
preDOS32DUMPPROCESS 00436 (0X01B4)  
preDOS32EXITLIST 00169 (0X00A9)  
preDOS32FREEMEM 00170 (0X00AA)  
preDOS32FREERESOURCE 00331 (0X014B)  
preDOS32GETNAMEDSHAREDMEM 00171 (0X00AB)  
preDOS32GETRESOURCE 00327 (0X0147)  
preDOS32GETSHAREDMEM 00172 (0X00AC)  
preDOS32GIVESHAREDMEM 00173 (0X00AD)  
preDOS32IASYNCTIMER 00310 (0X0136)  
preDOS32INITIALIZEPORTHOLE 00416 (0X01A0)  
preDOS32IPROTECTQUERYFHSTATE 00444 (0X01BC)  
preDOS32IPROTECTREAD 00508 (0X01FC)  
preDOS32IPROTECTSETFHSTATE 00442 (0X01BA)  
preDOS32IPROTECTWRITE 00510 (0X01FE)  
preDOS32IQUERYFHSTATE 00430 (0X01AE)  
preDOS32IREAD 00432 (0X01B0)  
preDOS32ISETFHSTATE 00428 (0X01AC)  
preDOS32ISTARTTIMER 00311 (0X0137)  
preDOS32IWRITE 00434 (0X01B2)  
preDOS32KILLTHREAD 00440 (0X01B8)  
preDOS32OPENEVENTSEM 00337 (0X0151)  
preDOS32OPENMUTEXSEM 00351 (0X015F)  
preDOS32OPENMUXWAITSEM 00363 (0X016B)  
preDOS32PMPOSTEVENTSEM 00482 (0X01E2)  
preDOS32PMWAITEVENTSEM 00493 (0X01ED)  
preDOS32PMWAITMUXWAITSEM 00485 (0X01E5)  
preDOS32POSTEVENTSEM 00343 (0X0157)  
preDOS32PROTECTSETFILELOCKS 00498 (0X01F2)



preDOS32QUERYEVENTSEM 00347 (0X015B)  
preDOS32QUERYEXTLIBPATH 00487 (0X01E7)  
preDOS32QUERYHEADERINFO 00418 (0X01A2)  
preDOS32QUERYMEM 00174 (0X00AE)  
preDOS32QUERYMEMSTATE 00175 (0X00AF)  
preDOS32QUERYMUTEXSEM 00359 (0X0167)  
preDOS32QUERYMUXWAITSEM 00373 (0X0175)  
preDOS32QUERYPROCADDR 00333 (0X014D)  
preDOS32QUERYPROCTYPE 00420 (0X01A4)  
preDOS32QUERYRESOURCESIZE 00410 (0X019A)  
preDOS32RELEASEMUTEXSEM 00357 (0X0165)  
preDOS32REQUESTMUTEXSEM 00355 (0X0163)  
preDOS32RESETEVENTSEM 00341 (0X0155)  
preDOS32SETEXTLIBPATH 00489 (0X01E9)  
preDOS32SETFILELOCKS 00496 (0X01F0)  
preDOS32SETMEM 00176 (0X00B0)  
preDOS32STARTTIMER 00379 (0X017B)  
preDOS32STOPTIMER 00312 (0X0138)  
preDOS32SUPPRESSPOPUIS 00438 (0X01B6)  
preDOS32VERIFYPIDTID 00491 (0X01EB)  
preDOS32WAITEVENTSEM 00345 (0X0159)  
preDOS32WAITMUXWAITSEM 00367 (0X016F)  
preDOS32WAITTHREAD 00391 (0X0187)  
preDOSACQUIRESPINLOCK 00502 (0X01F6)  
preDOSALLOCCHUGE 00177 (0X00B1)  
preDOSALLOCPROTHUGE 00178 (0X00B2)  
preDOSALLOCPROTSEG 00179 (0X00B3)  
preDOSALLOCSEG 00180 (0X00B4)  
preDOSALLOCSHRPROTSEG 00181 (0X00B5)  
preDOSALLOCSHRSEG 00182 (0X00B6)  
preDOSBEEP 00183 (0X00B7)  
preDOSBUFRESET 00184 (0X00B8)  
preDOSCALLNMPIPE 00185 (0X00B9)  
preDOSCHDIR 00186 (0X00BA)  
preDOSCHGFILEPTR 00187 (0X00BB)  
preDOSCLIACCESS 00188 (0X00BC)  
preDOSCLOSE 00189 (0X00BD)  
preDOSCLOSECHANGENOTIFY 00450 (0X01C2)  
preDOSCLOSESEM 00190 (0X00BE)  
preDOSCONNECTNMPIPE 00191 (0X00BF)  
preDOSCREATECSALIAS 00192 (0X00C0)  
preDOSCREATESEM 00193 (0X00C1)  
preDOSCREATESPINLOCK 00500 (0X01F4)  
preDOSCREATETHREAD 00194 (0X00C2)  
preDOSCWAIT 00195 (0X00C3)  
preDOSDELETE 00196 (0X00C4)  
preDOSDEVCONFIG 00197 (0X00C5)  
preDOSDEVIOCTL 00199 (0X00C7)  
preDOSDEVIOCTL2 00198 (0X00C6)  
preDOSDISCONNECTNMPIPE 00200 (0X00C8)  
preDOSDUPHANDLE 00201 (0X00C9)  
preDOSEDITNAME 00202 (0X00CA)  
preDOSENTERCRITSEC 00203 (0X00CB)  
preDOSENUMATTRIBUTE 00404 (0X0194)  
preDOSERROR 00204 (0X00CC)  
preDOSEXIT 00206 (0X00CE)  
preDOSEXITCRITSEC 00207 (0X00CF)  
preDOSEXITLIST 00208 (0X00D0)  
preDOSFILEIO 00209 (0X00D1)  
preDOSFILELOCKS 00210 (0X00D2)  
preDOSFINDCLOSE 00211 (0X00D3)  
preDOSFINDFIRST 00213 (0X00D5)  
preDOSFINDFIRST2 00212 (0X00D4)  
preDOSFINDFROMNAME 00214 (0X00D6)  
preDOSFINDNEXT 00215 (0X00D7)  
preDOSFINDNOTIFYCLOSE 00216 (0X00D8)  
preDOSFINDNOTIFYFIRST 00217 (0X00D9)  
preDOSFINDNOTIFYNEXT 00218 (0X00DA)  
preDOSFLAGPROCESS 00219 (0X00DB)  
preDOSFORCEDELETE 00458 (0X01CA)  
preDOSFREEMODULE 00220 (0X00DC)  
preDOSFREERESOURCE 00325 (0X0145)  
preDOSFREESEG 00221 (0X00DD)

preDOSFREESPINLOCK 00506 (0X01FA)  
preDOSFSATTACH 00222 (0X00DE)  
preDOSFSCTL 00223 (0X00DF)  
preDOSGETCP 00375 (0X0177)  
preDOSGETDATETIME 00224 (0X00E0)  
preDOSGETMODHANDLE 00225 (0X00E1)  
preDOSGETMODNAME 00226 (0X00E2)  
preDOSGETPID 00227 (0X00E3)  
preDOSGETPROCADDR 00228 (0X00E4)  
preDOSGETPRTY 00229 (0X00E5)  
preDOSGETRESOURCE 00230 (0X00E6)  
preDOSGETRESOURCE2 00323 (0X0143)  
preDOSGETSEG 00231 (0X00E7)  
preDOSGETSHRSEG 00232 (0X00E8)  
preDOSGETVERSION 00233 (0X00E9)  
preDOSGIVESEG 00234 (0X00EA)  
preDOSHOLD SIGNAL 00235 (0X00EB)  
preDOSICOPY 00236 (0X00EC)  
preDOSIDEVIOCTL 00412 (0X019C)  
preDOSIEXEC PGM 00237 (0X00ED)  
preDOSIPROTECTREAD 00460 (0X01CC)  
preDOSIPROTECTSETFILEINFO 00462 (0X01CE)  
preDOSIPROTECTWRITE 00464 (0X01D0)  
preDOSIREAD 00280 (0X0118)  
preDOSISETCP 00240 (0X00F0)  
preDOSISETFILEINFO 00406 (0X0196)  
preDOSISETPATHINFO 00408 (0X0198)  
preDOSISETRELMAXFH 00414 (0X019E)  
preDOSIWRITE 00315 (0X013B)  
preDOSKILLPROCESS 00241 (0X00F1)  
preDOSLOADMODULE 00242 (0X00F2)  
preDOSMAKENMPIPE 00243 (0X00F3)  
preDOSMAKEPIPE 00244 (0X00F4)  
preDOSMKDIR 00246 (0X00F6)  
preDOSMKDIR2 00245 (0X00F5)  
preDOSMOVE 00252 (0X00FC)  
preDOSNEWSIZE 00254 (0X00FE)  
preDOSOPEN 00256 (0X0100)  
preDOSOPEN2 00255 (0X00FF)  
preDOSOPEN2COMPT 00424 (0X01A8)  
preDOSOPENCHANGENOTIFY 00470 (0X01D6)  
preDOSOPENSEM 00257 (0X0101)  
preDOSOPLOCKRELEASE 00258 (0X0102)  
preDOSOPLOCKWAIT 00259 (0X0103)  
preDOSPEEKNMPIPE 00260 (0X0104)  
preDOSPHYSICALDISK 00261 (0X0105)  
preDOSPORTACCESS 00263 (0X0107)  
preDOSPROTECTCHGFILEPTR 00446 (0X01BE)  
preDOSPROTECTCLOSE 00448 (0X01C0)  
preDOSPROTECTENUMATTRIBUTE 00452 (0X01C4)  
preDOSPROTECTFILEIO 00454 (0X01C6)  
preDOSPROTECTFILELOCKS 00456 (0X01C8)  
preDOSPROTECTNEWSIZE 00466 (0X01D2)  
preDOSPROTECTOPEN 00468 (0X01D4)  
preDOSPROTECTQFHANDSTATE 00472 (0X01D8)  
preDOSPROTECTQFILEINFO 00474 (0X01DA)  
preDOSPROTECTSETFHANDSTATE 00478 (0X01DE)  
preDOSPROTECTSETFILEINFO 00480 (0X01E0)  
preDOSQCURDIR 00265 (0X0109)  
preDOSQCURDISK 00266 (0X010A)  
preDOSQFHANDSTATE 00267 (0X010B)  
preDOSQFILEINFO 00268 (0X010C)  
preDOSQFILEMODE 00269 (0X010D)  
preDOSQFSATTACH 00270 (0X010E)  
preDOSQFSINFO 00271 (0X010F)  
preDOSQHANDTYPE 00272 (0X0110)  
preDOSQNMPHANDSTATE 00273 (0X0111)  
preDOSQNMPIPEINFO 00274 (0X0112)  
preDOSQNMPIPESEMSTATE 00262 (0X0106)  
preDOSQPATHINFO 00275 (0X0113)  
preDOSQSYSINFO 00276 (0X0114)  
preDOSQVERIFY 00277 (0X0115)  
preDOSRAWREADNMPIPE 00278 (0X0116)



preDOSRAWWRITENMPIPE 00279 (0X0117)  
preDOSREALLOCHUGE 00282 (0X011A)  
preDOSREALLOCSEG 00283 (0X011B)  
preDOSRELEASESPINLOCK 00504 (0X01F8)  
preDOSRESETCHANGENOTIFY 00476 (0X01DC)  
preDOSRESUMETHREAD 00284 (0X011C)  
preDOSRMDIR 00285 (0X011D)  
preDOSSELECTDISK 00286 (0X011E)  
preDOSSEMSETWAIT 00290 (0X0122)  
preDOSSEENDSIGNAL 00292 (0X0124)  
preDOSSETDATETIME 00294 (0X0126)  
preDOSSETFHANDSTATE 00295 (0X0127)  
preDOSSETFILEINFO 00296 (0X0128)  
preDOSSETFILEMODE 00297 (0X0129)  
preDOSSETFSINFO 00298 (0X012A)  
preDOSSETMAXFH 00299 (0X012B)  
preDOSSETNMPHANDSTATE 00300 (0X012C)  
preDOSSETNMPIPESEM 00301 (0X012D)  
preDOSSETPATHINFO 00302 (0X012E)  
preDOSSETPRTY 00303 (0X012F)  
preDOSSETSIGHANDLER 00304 (0X0130)  
preDOSSETTRACEINFO 32768 (0X8000)  
preDOSSETVEC 00305 (0X0131)  
preDOSSETVERIFY 00306 (0X0132)  
preDOSSLEEP 00307 (0X0133)  
preDOSSUSPENDTHREAD 00308 (0X0134)  
preDOSSYSTEMSERVICE 00309 (0X0135)  
preDOSTRANSACTNMPIPE 00313 (0X0139)  
preDOSWAITNMPIPE 00314 (0X013A)  
preRelease 00395 (0X018B)  
w\_LockOper 00397 (0X018D)

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 1 (0X0001)

### Description

(OS) DosEnterCritSec Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.post2DOSENTERCITSEC

### Minor Code

1 (0X0001)

### Trace Groups

TK

### Trace Types

POST

### Traced Parameters

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 2 (0X0002)

### Description

(OS) DosExitCritSec Post-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.post2DOSEXITCRITSEC
<u>Minor Code</u>	2 (0X0002)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 3 (0X0003)

<u>Description</u>	(OS) DosHoldSignal Post-Invocation (2)
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.post2DOSHOLDSIGNAL
<u>Minor Code</u>	3 (0X0003)
<u>Trace Groups</u>	SIG
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 6 (0X0006)

<u>Description</u>	(OS) Dos32AliasMem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALIASMEM
<u>Minor Code</u>	6 (0X0006)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	

Alias address = %F Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 7 (0X0007)

### Description

(OS) Dos32AllocMem Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALLOCMEM

### Minor Code

7 (0X0007)

### Trace Groups

VM

### Trace Types

POST

### Traced Parameters

Address = %F Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 8 (0X0008)

### Description

(OS) Dos32AllocProtectedMem Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALLOCPROTECTEDMEM

### Minor Code

8 (0X0008)

### Trace Groups

VM

### Trace Types

POST

### Traced Parameters

Address = %F Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 9 (0X0009)

### Description

(OS) Dos32AllocSharedMem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ALLOCSharedMEM

**Minor Code**

9 (0X0009)

**Trace Groups**

VM

**Trace Types**

POST

**Traced Parameters**

Address = %F Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 10 (0X000A)

**Description**

(OS) Dos32CreateThread Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATETHREAD

**Minor Code**

10 (0X000A)

**Trace Groups**

TK

**Trace Types**

POST

**Traced Parameters**

Thread ID = %D Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 11 (0X000B)

**Description**

(OS) Dos32Debug Dos Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32DEBUG

**Minor Code**

11 (0X000B)

**Trace Groups**

TK

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 12 (0X000C)

**Description**

(OS) Dos32ExitList Dos Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32EXITLIST

**Minor Code**

12 (0X000C)

**Trace Groups**

TK

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 13 (0X000D)

**Description**

(OS) Dos32FreeMem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32FREEMEM

**Minor Code**

13 (0X000D)

**Trace Groups**

VM

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 14 (0X000E)

<u>Description</u>	(OS) Dos32GetNamedSharedMem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GETNAMEDSHAREDMEM
<u>Minor Code</u>	14 (0X000E)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Address = %F Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 15 (0X000F)

<u>Description</u>	(OS) Dos32GetSharedMem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GETSHAREDMEM
<u>Minor Code</u>	15 (0X000F)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 16 (0X0010)

<u>Description</u>	(OS) Dos32GiveSharedMem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GIVESHAREDMEM
<u>Minor Code</u>	16 (0X0010)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 17 (0X0011)

**Description**

(OS) Dos32QueryMem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMEM

**Minor Code**

17 (0X0011)

**Trace Groups**

VM

**Trace Types**

POST

**Traced Parameters**

Actual size = %D Flags = %D

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 18 (0X0012)

**Description**

(OS) Dos32QueryMemState Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMEMSTATE

**Minor Code**

18 (0X0012)

**Trace Groups**

VM

**Trace Types**

POST

**Traced Parameters**

Actual size = %D Flags = %D

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 19 (0X0013)

<u><b>Description</b></u>	(OS) Dos32SetMem Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SETMEM
<u><b>Minor Code</b></u>	19 (0X0013)
<u><b>Trace Groups</b></u>	VM
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	Return code = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 20 (0X0014)

<u><b>Description</b></u>	(OS) DosAllocHuge Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCHUGE
<u><b>Minor Code</b></u>	20 (0X0014)
<u><b>Trace Groups</b></u>	SEL
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	Base selector = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 21 (0X0015)

<u><b>Description</b></u>	(OS) DosAllocProtHuge Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCPROTHUGE
<u><b>Minor Code</b></u>	21 (0X0015)
<u><b>Trace Groups</b></u>	SEL



Trace Types  
POST

Traced Parameters  
  
Base selector = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 22 (0X0016)

Description  
(OS) DosAllocProtSeg Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCROTSEG

Minor Code  
22 (0X0016)

Trace Groups  
SEL

Trace Types  
POST

Traced Parameters  
  
Selector = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 23 (0X0017)

Description  
(OS) DosAllocSeg Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCSEG

Minor Code  
23 (0X0017)

Trace Groups  
SEL

Trace Types  
POST

Traced Parameters  
  
Selector = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 24 (0X0018)

<u><b>Description</b></u>	(OS) DosAllocShrProtSeg Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCshrPROTSEG
<u><b>Minor Code</b></u>	24 (0X0018)
<u><b>Trace Groups</b></u>	SEL
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	Selector = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 25 (0X0019)

<u><b>Description</b></u>	(OS) DosAllocShrSeg Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSALLOCshrSEG
<u><b>Minor Code</b></u>	25 (0X0019)
<u><b>Trace Groups</b></u>	SEL
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	Selector = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 26 (0X001A)

<u><b>Description</b></u>	(OS) DosBeep Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSBEEP
<u><b>Minor Code</b></u>	26 (0X001A)
<u><b>Trace Groups</b></u>	IO

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 27 (0X001B)

**Description**

(OS) DosBufReset Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSBUFRESET

**Minor Code**

27 (0X001B)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 28 (0X001C)

**Description**

(OS) DosCallNmPipe Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCALLNMPIPE

**Minor Code**

28 (0X001C)

**Trace Groups**

PIP

**Trace Types**

POST

**Traced Parameters**

Bytes out = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 29 (0X001D)

<u>Description</u>	(OS) DosChDir Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCHDIR
<u>Minor Code</u>	29 (0X001D)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 30 (0X001E)

<u>Description</u>	(OS) DosChgFilePtr Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCHGFILEPTR
<u>Minor Code</u>	30 (0X001E)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Location = %W%W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 31 (0X001F)

<u>Description</u>	(OS) DosCliAccess Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLIACCESS
<u>Minor Code</u>	31 (0X001F)
<u>Trace Groups</u>	TK

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 32 (0X0020)

Description  
(OS) DosClose Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLOSE

Minor Code  
32 (0X0020)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 33 (0X0021)

Description  
(OS) DosCloseSem Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLOSESEM

Minor Code  
33 (0X0021)

Trace Groups  
SEM

Trace Types  
POST

Traced Parameters  
  
Return value = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 34 (0X0022)

<b>Description</b>	(OS) DosConnectNmPipe Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCONNECTNMPIPE
<b>Minor Code</b>	34 (0X0022)
<b>Trace Groups</b>	PIP
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 35 (0X0023)

<b>Description</b>	(OS) DosCreateCSAlias Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATECSALIAS
<b>Minor Code</b>	35 (0X0023)
<b>Trace Groups</b>	SEL
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Code selector = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 36 (0X0024)

<b>Description</b>	(OS) DosCreateSem Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATESEM
<b>Minor Code</b>	36 (0X0024)
<b>Trace Groups</b>	SEL

Trace Types  
POST

Traced Parameters  
  
Handle = %W%W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 37 (0X0025)

Description  
(OS) DosCreateThread Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATETHREAD

Minor Code  
37 (0X0025)

Trace Groups  
TK

Trace Types  
POST

Traced Parameters  
  
Thread ID = %W Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 38 (0X0026)

Description  
(OS) DosCWait Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCWAIT

Minor Code  
38 (0X0026)

Trace Groups  
TK

Trace Types  
POST

Traced Parameters  
  
Pid = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 39 (0X0027)

<u>Description</u>	(OS) DosDelete Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDELETE
<u>Minor Code</u>	39 (0X0027)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 40 (0X0028)

<u>Description</u>	(OS) DosDevConfig Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDEVCONFIG
<u>Minor Code</u>	40 (0X0028)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 41 (0X0029)

<u>Description</u>	(OS) DosDevIoctl2 Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDEVIOCTL2
<u>Minor Code</u>	41 (0X0029)
<u>Trace Groups</u>	FS



**Trace Types** POST

**Traced Parameters**  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 42 (0X002A)

**Description** (OS) DosDevIoctl Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDEVIOCTL

**Minor Code** 42 (0X002A)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 43 (0X002B)

**Description** (OS) DosDisConnectMmPipe Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDISCONNECTNMPIPE

**Minor Code** 43 (0X002B)

**Trace Groups** PIP

**Trace Types** POST

**Traced Parameters**  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 44 (0X002C)

<u>Description</u>	(OS) DosDupHandle Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSDUPHANDLE
<u>Minor Code</u>	44 (0X002C)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	New handle = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 45 (0X002D)

<u>Description</u>	(OS) DosEditName Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSEDITNAME
<u>Minor Code</u>	45 (0X002D)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Resultant string = %S Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 47 (0X002F)

<u>Description</u>	(OS) DosError Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSError
<u>Minor Code</u>	47 (0X002F)

**Trace Groups** TK

**Trace Types** POST

**Traced Parameters**

Return value = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 49 (0X0031)

**Description** (OS) Dos32Exit Dos Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSEXIT

**Minor Code** 49 (0X0031)

**Trace Groups** TK

**Trace Types** POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 51 (0X0033)

**Description** (OS) Dos32ExitList Dos Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSEXITLIST

**Minor Code** 51 (0X0033)

**Trace Groups** TK

**Trace Types** POST

**Traced Parameters**

Return code = %D

# OS2KRNL Major Code: 0X0005 Minor Code: 52 (0X0034)

<u>Description</u>	(OS) DosFileIO Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFILEIO
<u>Minor Code</u>	52 (0X0034)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 53 (0X0035)

<u>Description</u>	(OS) DosFileLocks Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFILELOCKS
<u>Minor Code</u>	53 (0X0035)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 54 (0X0036)

<u>Description</u>	(OS) DosFindClose Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDCLOSE
<u>Minor Code</u>	54 (0X0036)

Trace Groups FS

Trace Types POST

Traced Parameters

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 55 (0X0037)

Description (OS) DosFindFirst2 Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDFIRST2

Minor Code 55 (0X0037)

Trace Groups FS

Trace Types POST

Traced Parameters

Search handle = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 56 (0X0038)

Description (OS) DosFindFirst Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDFIRST

Minor Code 56 (0X0038)

Trace Groups FS

Trace Types POST

Traced Parameters

Search handle = %W Return code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 57 (0X0039)

Description	(OS) DosFindFromName Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDFROMNAME
Minor Code	57 (0X0039)
Trace Groups	FS
Trace Types	POST
Traced Parameters	Search count = %W Return code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 58 (0X003A)

Description	(OS) DosFindNext Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNEXT
Minor Code	58 (0X003A)
Trace Groups	FS
Trace Types	POST
Traced Parameters	Return code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 59 (0X003B)

Description	(OS) DosFindNotifyClose Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNOTIFYCLOSE
Minor Code	

59 (0X003B)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 60 (0X003C)

**Description**

(OS) DosFindNotifyFirst Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNOTIFYFIRST

**Minor Code**

60 (0X003C)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Search count = %W Handle = %W

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 61 (0X003D)

**Description**

(OS) DosFindNotifyNext Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFINDNOTIFYNEXT

**Minor Code**

61 (0X003D)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Change count = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 62 (0X003E)

<u>Description</u>	(OS) DosFlagProcess Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFLAGPROCESS
<u>Minor Code</u>	62 (0X003E)
<u>Trace Groups</u>	SIG
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 63 (0X003F)

<u>Description</u>	(OS) DosFreeModule Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREEMODULE
<u>Minor Code</u>	63 (0X003F)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 64 (0X0040)

<u>Description</u>	(OS) DosFreeSeg Post-Invocation
<u>Tracepoint</u>	



Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREESEG

**Minor Code**

64 (0X0040)

**Trace Groups**

SEL

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 65 (0X0041)

**Description**

(OS) DosFSAttach Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFSATTACH

**Minor Code**

65 (0X0041)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 66 (0X0042)

**Description**

(OS) DosFSctl Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFSCTL

**Minor Code**

66 (0X0042)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return code = %W

---

## OS2KRNL Major Code: 0X0005 Minor Code: 67 (0X0043)

**Description**

(OS) DosGetDateTime Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETDATETIME

**Minor Code**

67 (0X0043)

**Trace Groups**

TIM

**Trace Types**

POST

**Traced Parameters**

Return code = %W

---

## OS2KRNL Major Code: 0X0005 Minor Code: 68 (0X0044)

**Description**

(OS) DosGetModHandle Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETMODHANDLE

**Minor Code**

68 (0X0044)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Module handle = %W Return code = %W

---

## OS2KRNL Major Code: 0X0005 Minor Code: 69 (0X0045)

**Description**

(OS) DosGetModName Post-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETMODNAME
<u>Minor Code</u>	69 (0X0045)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Module name = %S
	Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 70 (0X0046)

<u>Description</u>	(OS) DosGetPid Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETPID
<u>Minor Code</u>	70 (0X0046)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	PID = %W TID = %W PPID = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 71 (0X0047)

<u>Description</u>	(OS) DosGetProcAddr Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETPROCADDR
<u>Minor Code</u>	71 (0X0047)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	POST

**Traced Parameters**

Proc addr = %W:%W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 72 (0X0048)

**Description**

(OS) DosGetPrty Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETPRTY

**Minor Code**

72 (0X0048)

**Trace Groups**

TK

**Trace Types**

POST

**Traced Parameters**

Priority = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 73 (0X0049)

**Description**

(OS) DosGetResource Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETRESOURCE

**Minor Code**

73 (0X0049)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Selector = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 74 (0X004A)

<b><u>Description</u></b>	(OS) DosGetSeg Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETSEG
<b><u>Minor Code</u></b>	74 (0X004A)
<b><u>Trace Groups</u></b>	SEL
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 75 (0X004B)

<b><u>Description</u></b>	(OS) DosGetShrSeg Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETSHRSEG
<b><u>Minor Code</u></b>	75 (0X004B)
<b><u>Trace Groups</u></b>	SEL
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Selector = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 76 (0X004C)

<b><u>Description</u></b>	(OS) DosGetVersion Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETVERSION
<b><u>Minor Code</u></b>	76 (0X004C)
<b><u>Trace Groups</u></b>	TK
<b><u>Trace Types</u></b>	POST

**Traced Parameters**

Major/minor version = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 77 (0X004D)

**Description**

(OS) DosGiveSeg Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGIVESEG

**Minor Code**

77 (0X004D)

**Trace Groups**

SEL

**Trace Types**

POST

**Traced Parameters**

Selector = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 78 (0X004E)

**Description**

(OS) DosHoldSignal Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSHOLD SIGNAL

**Minor Code**

78 (0X004E)

**Trace Groups**

SIG

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 79 (0X004F)

<b><u>Description</u></b>	(OS) DoslCopy Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSICOPY
<b><u>Minor Code</u></b>	79 (0X004F)
<b><u>Trace Groups</u></b>	FS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 80 (0X0050)

<b><u>Description</u></b>	(OS) DoslExecPgm Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIEXECPGM
<b><u>Minor Code</u></b>	80 (0X0050)
<b><u>Trace Groups</u></b>	TK
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Pid/Term Code = %W Result Code = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 83 (0X0053)

<b><u>Description</u></b>	(OS) DoslSetCP Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSISETCP
<b><u>Minor Code</u></b>	83 (0X0053)
<b><u>Trace Groups</u></b>	TK

**Trace Types**  
POST

**Traced Parameters**  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 84 (0X0054)

**Description**  
(OS) DosKillProcess Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSKILLPROCESS

**Minor Code**  
84 (0X0054)

**Trace Groups**  
SIG

**Trace Types**  
POST

**Traced Parameters**  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 85 (0X0055)

**Description**  
(OS) DosLoadModule Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSLOADMODULE

**Minor Code**  
85 (0X0055)

**Trace Groups**  
LDR

**Trace Types**  
POST

**Traced Parameters**  
  
Module handle = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 86 (0X0056)



<b>Description</b>	(OS) DosMakeNmPipe Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMAKENMPIPE
<b>Minor Code</b>	86 (0X0056)
<b>Trace Groups</b>	PIP
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Handle = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 87 (0X0057)

<b>Description</b>	(OS) DosMakePipe Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMAKEPIPE
<b>Minor Code</b>	87 (0X0057)
<b>Trace Groups</b>	PIP
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Read handle = %W Write handle = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 88 (0X0058)

<b>Description</b>	(OS) DosMkDir2 Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMKDIR2
<b>Minor Code</b>	88 (0X0058)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 89 (0X0059)

**Description** (OS) DosMkDir Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMKDIR

**Minor Code** 89 (0X0059)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 95 (0X005F)

**Description** (OS) DosMove Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMOVE

**Minor Code** 95 (0X005F)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return code = %W

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 96 (0X0060)

Description	(OS) DosMuxSemWait Post-Invocation (Waited)
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMUXSEMWAIT
Minor Code	96 (0X0060)
Trace Groups	SEM
Trace Types	PRE
Traced Parameters	Index = %W Return Code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 97 (0X0061)

Description	(OS) DosNewSize Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSNEWSIZE
Minor Code	97 (0X0061)
Trace Groups	FS
Trace Types	POST
Traced Parameters	Return Code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 98 (0X0062)

Description	(OS) DosOpen2 Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPEN2
Minor Code	98 (0X0062)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Action = %W Handle = %W

Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 99 (0X0063)

**Description**

(OS) DosOpen Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPEN

**Minor Code**

99 (0X0063)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Action = %W Handle = %W

Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 100 (0X0064)

**Description**

(OS) DosOpenSem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPENSEM

**Minor Code**

100 (0X0064)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Handle = %W%W Return code = %W

---

## OS2KRNL Major Code: 0X0005 Minor Code: 101 (0X0065)

**Description**

(OS) DosOpLockRelease Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPLOCKRELEASE

**Minor Code**

101 (0X0065)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

No parameters traced.

---

## OS2KRNL Major Code: 0X0005 Minor Code: 102 (0X0066)

**Description**

(OS) DosOpLockWait Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPLOCKWAIT

**Minor Code**

102 (0X0066)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Server Key = %D Kernel Key = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 103 (0X0067)

**Description**

(OS) DosPeekNmPipe Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPEEKNMPIPE

**Minor Code** 103 (0X0067)

**Trace Groups** PIP

**Trace Types** POST

**Traced Parameters**  
Bytes read = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 104 (0X0068)

**Description** (OS) DosPhysicalDisk Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPHYSICALDISK

**Minor Code** 104 (0X0068)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**  
Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 105 (0X0069)

**Description** (OS) DosQNmPipeState Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQNMPIPESEMSTATE

**Minor Code** 105 (0X0069)

**Trace Groups** PIP

**Trace Types** POST

**Traced Parameters**  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 106 (0X006A)

<u>Description</u>	(OS) DosPortAccess Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPORTACCESS
<u>Minor Code</u>	106 (0X006A)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 108 (0X006C)

<u>Description</u>	(OS) DosQCurDir Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQCURDIR
<u>Minor Code</u>	108 (0X006C)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Current Dir = %S Len = %W Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 109 (0X006D)

<u>Description</u>	(OS) DosQCurDisk Post-Invocation
--------------------	----------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQCURDISK
<u>Minor Code</u>	109 (0X006D)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	<p>Default Drive = %W Logical Map =%D</p> <p>Return Code = %W</p>

## OS2KRNL Major Code: 0X0005 Minor Code: 110 (0X006E)

<u>Description</u>	(OS) DosQFHandState Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFHANDSTATE
<u>Minor Code</u>	110 (0X006E)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	<p>Handle State = %W Return Code = %W</p>

## OS2KRNL Major Code: 0X0005 Minor Code: 111 (0X006F)

<u>Description</u>	(OS) DosQFileInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFILEINFO
<u>Minor Code</u>	111 (0X006F)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST



**Traced Parameters**

Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 112 (0X0070)

**Description**

(OS) DosQFileMode Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFILEMODE

**Minor Code**

112 (0X0070)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

File Mode = %W Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 113 (0X0071)

**Description**

(OS) DosQFSAttach Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFSATTACH

**Minor Code**

113 (0X0071)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 114 (0X0072)

<u>Description</u>	(OS) DosQFSInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQFSINFO
<u>Minor Code</u>	114 (0X0072)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 115 (0X0073)

<u>Description</u>	(OS) DosQHandType Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQHANDTYPE
<u>Minor Code</u>	115 (0X0073)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Handle Type = %W Flags = %W Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 116 (0X0074)

<u>Description</u>	(OS) DosQNMPHandState Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQNMPHANDSTATE
<u>Minor Code</u>	116 (0X0074)
<u>Trace Groups</u>	PIP

**Trace Types**  
POST

**Traced Parameters**  
  
Handle state = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 117 (0X0075)

**Description**  
(OS) DosQNmPipeInfo Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQNMPIPEINFO

**Minor Code**  
117 (0X0075)

**Trace Groups**  
PIP

**Trace Types**  
POST

**Traced Parameters**  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 118 (0X0076)

**Description**  
(OS) DosQPathInfo Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQPATHINFO

**Minor Code**  
118 (0X0076)

**Trace Groups**  
FS

**Trace Types**  
POST

**Traced Parameters**  
  
Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 119 (0X0077)

<u>Description</u>	(OS) DosQSysInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQSYSINFO
<u>Minor Code</u>	119 (0X0077)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 120 (0X0078)

<u>Description</u>	(OS) DosQVerify Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSQVERIFY
<u>Minor Code</u>	120 (0X0078)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Verify Flag = %W Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 121 (0X0079)

<u>Description</u>	(OS) DosRawReadNmPipe Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRAWREADNMPIPE
<u>Minor Code</u>	121 (0X0079)
<u>Trace Groups</u>	PIP

Trace Types  
POST

Traced Parameters  
  
Bytes read = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 122 (0X007A)

Description  
(OS) DosRawWriteNmPipe Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRAWWRITENMPIPE

Minor Code  
122 (0X007A)

Trace Groups  
PIP

Trace Types  
POST

Traced Parameters  
  
Bytes written = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 123 (0X007B)

Description  
(OS) DoslRead Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIREAD

Minor Code  
123 (0X007B)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
  
Bytes Read = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 125 (0X007D)

<u>Description</u>	(OS) DosReallocHuge Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSREALLOCHUGE
<u>Minor Code</u>	125 (0X007D)
<u>Trace Groups</u>	SEL
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 126 (0X007E)

<u>Description</u>	(OS) DosReallocSeg Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSREALLOCSEG
<u>Minor Code</u>	126 (0X007E)
<u>Trace Groups</u>	SEL
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 127 (0X007F)

<u>Description</u>	(OS) Dos32ResumeThread Dos Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRESUMETHREAD
<u>Minor Code</u>	127 (0X007F)
<u>Trace Groups</u>	TK

Trace Types POST

Traced Parameters  
Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 128 (0X0080)

Description (OS) DosRmdir Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRMDIR

Minor Code 128 (0X0080)

Trace Groups FS

Trace Types POST

Traced Parameters  
Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 129 (0X0081)

Description (OS) DosSelectDisk Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSELECTDISK

Minor Code 129 (0X0081)

Trace Groups FS

Trace Types POST

Traced Parameters  
Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 133 (0X0085)

<u>Description</u>	(OS) DosSemSetWait Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSEMSETWAIT
<u>Minor Code</u>	133 (0X0085)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 135 (0X0087)

<u>Description</u>	(OS) DosSendSignal Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSEND SIGNAL
<u>Minor Code</u>	135 (0X0087)
<u>Trace Groups</u>	SIG
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 137 (0X0089)

<u>Description</u>	(OS) DosSetDateTime Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETDATETIME
<u>Minor Code</u>	137 (0X0089)
<u>Trace Groups</u>	TIM



Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 138 (0X008A)

Description  
(OS) DosSetFHandState Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFHANDSTATE

Minor Code  
138 (0X008A)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
  
Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 139 (0X008B)

Description  
(OS) DosSetFileInfo Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFILEINFO

Minor Code  
139 (0X008B)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
  
Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 140 (0X008C)

<u>Description</u>	(OS) DosSetFileMode Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFILEMODE
<u>Minor Code</u>	140 (0X008C)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 141 (0X008D)

<u>Description</u>	(OS) DosSetFSInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETFSINFO
<u>Minor Code</u>	141 (0X008D)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 142 (0X008E)

<u>Description</u>	(OS) DosSetMaxFH Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETMAXFH
<u>Minor Code</u>	142 (0X008E)
<u>Trace Groups</u>	FS

Trace Types  
POST

Traced Parameters  
  
Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 143 (0X008F)

Description  
(OS) DosSetNmPHandState Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETNMPHANDSTATE

Minor Code  
143 (0X008F)

Trace Groups  
PIP

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 144 (0X0090)

Description  
(OS) DosSetNmPipeSem Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETNMPIPESEM

Minor Code  
144 (0X0090)

Trace Groups  
PIP

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 145 (0X0091)

<u>Description</u>	(OS) DosSetPathInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETPATHINFO
<u>Minor Code</u>	145 (0X0091)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 146 (0X0092)

<u>Description</u>	(OS) DosSetPrty Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETPRTY
<u>Minor Code</u>	146 (0X0092)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 147 (0X0093)

<u>Description</u>	(OS) DosSetSigHandler Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETSIGHANDLER
<u>Minor Code</u>	147 (0X0093)
<u>Trace Groups</u>	SIG

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 148 (0X0094)

Description  
(OS) DosSetVec Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETVEC

Minor Code  
148 (0X0094)

Trace Groups  
SIG

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 149 (0X0095)

Description  
(OS) DosSetVerify Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETVERIFY

Minor Code  
149 (0X0095)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
  
Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 150 (0X0096)

<u>Description</u>	(OS) DosSleep Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSLEEP
<u>Minor Code</u>	150 (0X0096)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 151 (0X0097)

<u>Description</u>	(OS) Dos32SuspendThread Dos Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSUSPENDTHREAD
<u>Minor Code</u>	151 (0X0097)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 152 (0X0098)

<u>Description</u>	(OS) DosSystemService Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSYSTEMSERVICE
<u>Minor Code</u>	152 (0X0098)
<u>Trace Groups</u>	TK

Trace Types  
POST

Traced Parameters  
  
Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 153 (0X0099)

Description  
(OS) DosTimerAsync Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IASYNCTIMER

Minor Code  
153 (0X0099)

Trace Groups  
TIM

Trace Types  
POST

Traced Parameters  
  
Return code = %W Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 154 (0X009A)

Description  
(OS) DosTimerStart Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ISTARTTIMER

Minor Code  
154 (0X009A)

Trace Groups  
TIM

Trace Types  
POST

Traced Parameters  
  
Return code = %W Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 155 (0X009B)

<u><b>Description</b></u>	(OS) Dos32StopTimer Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32STOPTIMER
<u><b>Minor Code</b></u>	155 (0X009B)
<u><b>Trace Groups</b></u>	TIM
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	Return code = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 156 (0X009C)

<u><b>Description</b></u>	(OS) DosTransactNmPipe Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSTRANSACTNMPIPE
<u><b>Minor Code</b></u>	156 (0X009C)
<u><b>Trace Groups</b></u>	PIP
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	Bytes out = %W Return code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 157 (0X009D)

<u><b>Description</b></u>	(OS) DosWaitNmPipe Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSWAITNMPIPE
<u><b>Minor Code</b></u>	157 (0X009D)
<u><b>Trace Groups</b></u>	PIP



**Trace Types**  
POST

**Traced Parameters**  
  
Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 158 (0X009E)

**Description**  
(OS) DosIWrite Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIWRITE

**Minor Code**  
158 (0X009E)

**Trace Groups**  
FS

**Trace Types**  
POST

**Traced Parameters**  
  
Bytes Written = %W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 163 (0X00A3)

**Description**  
(OS) Dos32AliasMem Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALIASEMEM

**Minor Code**  
163 (0X00A3)

**Trace Groups**  
VM

**Trace Types**  
PRE

**Traced Parameters**  
  
Address = %F Size = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 164 (0X00A4)

<u>Description</u>	(OS) Dos32AllocMem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALLOCMEM
<u>Minor Code</u>	164 (0X00A4)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	<p>Address = %F Size = %D</p> <p>Flags = %D</p>

## OS2KRNL Major Code: 0X0005 Minor Code: 165 (0X00A5)

<u>Description</u>	(OS) Dos32AllocProtectedMem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALLOCPROTECTEDMEM
<u>Minor Code</u>	165 (0X00A5)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	<p>Size = %D Flags = %D</p>

## OS2KRNL Major Code: 0X0005 Minor Code: 166 (0X00A6)

<u>Description</u>	(OS) Dos32AllocSharedMem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ALLOCSHAREDMEM
<u>Minor Code</u>	166 (0X00A6)

**Trace Groups** VM

**Trace Types** PRE

**Traced Parameters**

Size = %D Flags = %D

OS2KRNL Major Code: 0X0005 Minor Code: 167 (0X00A7)

**Description** (OS) Dos32CreateThread Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATETHREAD

**Minor Code** 167 (0X00A7)

**Trace Groups** TK

**Trace Types** PRE

**Traced Parameters**

Stack size = %D Flags = %D

Arg pointer = %F Starting EIP = %F

OS2KRNL Major Code: 0X0005 Minor Code: 168 (0X00A8)

**Description** (OS) Dos32Debug Dos Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32DEBUG

**Minor Code** 168 (0X00A8)

**Trace Groups** TK

**Trace Types** PRE

**Traced Parameters** No parameters traced.

# OS2KRNL Major Code: 0X0005 Minor Code: 169 (0X00A9)

<u>Description</u>	(OS) Dos32ExitList Dos Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32EXITLIST
<u>Minor Code</u>	169 (0X00A9)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Function = %D    Address = %F

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 170 (0X00AA)

<u>Description</u>	(OS) Dos32FreeMem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32FREEMEM
<u>Minor Code</u>	170 (0X00AA)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Address = %F

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 171 (0X00AB)

<u>Description</u>	(OS) Dos32GetNamedSharedMem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GETNAMEDSHAREDMEM
<u>Minor Code</u>	171 (0X00AB)

Trace Groups  
VM

Trace Types  
PRE

Traced Parameters  
  
Name = %S  
Flags = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 172 (0X00AC)

Description  
(OS) Dos32GetSharedMem Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GETSHAREDMEM

Minor Code  
172 (0X00AC)

Trace Groups  
VM

Trace Types  
PRE

Traced Parameters  
  
Address = %F Flags = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 173 (0X00AD)

Description  
(OS) Dos32GiveSharedMem Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GIVESHAREDMEM

Minor Code  
173 (0X00AD)

Trace Groups  
VM

Trace Types  
PRE

Traced Parameters  
  
Address = %F Process ID = %D

Flags = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 174 (0X00AE)

### Description

(OS) Dos32QueryMem Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYMEM

### Minor Code

174 (0X00AE)

### Trace Groups

VM

### Trace Types

PRE

### Traced Parameters

Address = %F Size = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 175 (0X00AF)

### Description

(OS) Dos32QueryMemState Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYMEMSTATE

### Minor Code

175 (0X00AF)

### Trace Groups

VM

### Trace Types

PRE

### Traced Parameters

Address = %F Size = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 176 (0X00B0)

### Description

(OS) Dos32SetMem Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SETMEM
<b><u>Minor Code</u></b>	176 (0X00B0)
<b><u>Trace Groups</u></b>	VM
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>Address = %F Size = %D</p> <p>Flags = %D</p>

## OS2KRNL Major Code: 0X0005 Minor Code: 177 (0X00B1)

<b><u>Description</u></b>	(OS) DosAllocHuge Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCHUGE
<b><u>Minor Code</u></b>	177 (0X00B1)
<b><u>Trace Groups</u></b>	SEL
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>Initial size=%W%W Max size = %W0000</p>

## OS2KRNL Major Code: 0X0005 Minor Code: 178 (0X00B2)

<b><u>Description</u></b>	(OS) DosAllocProtHuge Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCPROTHUGE
<b><u>Minor Code</u></b>	178 (0X00B2)
<b><u>Trace Groups</u></b>	SEL
<b><u>Trace Types</u></b>	PRE

**Traced Parameters**

Initial size=%W%W Max size = %W0000

-----

OS2KRNL Major Code: 0X0005 Minor Code: 179 (0X00B3)

**Description**

(OS) DosAllocProtSeg Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCPROTSEG

**Minor Code**

179 (0X00B3)

**Trace Groups**

SEL

**Trace Types**

PRE

**Traced Parameters**

Flags=%W Size=%W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 180 (0X00B4)

**Description**

(OS) DosAllocSeg Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCSEG

**Minor Code**

180 (0X00B4)

**Trace Groups**

SEL

**Trace Types**

PRE

**Traced Parameters**

Flags=%W Size=%W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 181 (0X00B5)



<u>Description</u>	(OS) DosAllocShrProtSeg Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCshrPROTSEG
<u>Minor Code</u>	181 (0X00B5)
<u>Trace Groups</u>	SEL
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Name = %S
	Size = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 182 (0X00B6)

<u>Description</u>	(OS) DosAllocShrSeg Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSALLOCshrSEG
<u>Minor Code</u>	182 (0X00B6)
<u>Trace Groups</u>	SEL
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Name = %S
	Size = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 183 (0X00B7)

<u>Description</u>	(OS) DosBeep Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSBEEP
<u>Minor Code</u>	183 (0X00B7)
<u>Trace Groups</u>	

IO

**Trace Types**

PRE

**Traced Parameters**

No parameters traced.

-----

OS2KRNL Major Code: 0X0005 Minor Code: 184 (0X00B8)

**Description**

(OS) DosBufReset Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSBUFRESET

**Minor Code**

184 (0X00B8)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 185 (0X00B9)

**Description**

(OS) DosCallNmPipe Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCALLNMPIPE

**Minor Code**

185 (0X00B9)

**Trace Groups**

PIP

**Trace Types**

PRE

**Traced Parameters**

Name = %S

-----

OS2KRNL Major Code: 0X0005 Minor Code: 186 (0X00BA)

<u><b>Description</b></u>	(OS) DosChDir Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCHDIR
<u><b>Minor Code</b></u>	186 (0X00BA)
<u><b>Trace Groups</b></u>	FS
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Path = %S

## OS2KRNL Major Code: 0X0005 Minor Code: 187 (0X00BB)

<u><b>Description</b></u>	(OS) DosChgFilePtr Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCHGFILEPTR
<u><b>Minor Code</b></u>	187 (0X00BB)
<u><b>Trace Groups</b></u>	FS
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Type = %W Distance = %W%W Handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 188 (0X00BC)

<u><b>Description</b></u>	(OS) DosCliAccess Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLIACCESS
<u><b>Minor Code</b></u>	188 (0X00BC)

<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	No parameters traced.

-----

OS2KRNL Major Code: 0X0005 Minor Code: 189 (0X00BD)

<u>Description</u>	(OS) DosClose Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLOSE
<u>Minor Code</u>	189 (0X00BD)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 190 (0X00BE)

<u>Description</u>	(OS) DosCloseSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLOSESEM
<u>Minor Code</u>	190 (0X00BE)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Semaphore handle = %W%W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 191 (0X00BF)

<u>Description</u>	(OS) DosConnectNmPipe Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCONNECTNMPIPE
<u>Minor Code</u>	191 (0X00BF)
<u>Trace Groups</u>	PIP
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 192 (0X00C0)

<u>Description</u>	(OS) DosCreateCSAlias Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATECSALIAS
<u>Minor Code</u>	192 (0X00C0)
<u>Trace Groups</u>	SEL
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Data selector = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 193 (0X00C1)

<u>Description</u>	(OS) DosCreateSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATESEM
<u>Minor Code</u>	193 (0X00C1)
<u>Trace Groups</u>	SEM

<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Name = %S
	-----

## OS2KRNL Major Code: 0X0005 Minor Code: 194 (0X00C2)

<u>Description</u>	(OS) DosCreateThread Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATETHREAD
<u>Minor Code</u>	194 (0X00C2)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Transfer Address = %A Stack Address = %A
	-----

## OS2KRNL Major Code: 0X0005 Minor Code: 195 (0X00C3)

<u>Description</u>	(OS) DosCWait Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCWAIT
<u>Minor Code</u>	195 (0X00C3)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Pid = %W Wait = %W Action = %W
	-----

## OS2KRNL Major Code: 0X0005 Minor Code: 196 (0X00C4)

<u>Description</u>	(OS) DosDelete Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDELETE
<u>Minor Code</u>	196 (0X00C4)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Path = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 197 (0X00C5)

<u>Description</u>	(OS) DosDevConfig Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDEVCONFIG
<u>Minor Code</u>	197 (0X00C5)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Parm = %W Item = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 198 (0X00C6)

<u>Description</u>	(OS) DosDevIoctl2 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDEVIOCTL2
<u>Minor Code</u>	198 (0X00C6)
<u>Trace Groups</u>	FS

<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %W Category = %W Function = %W
	-----

OS2KRNL Major Code: 0X0005 Minor Code: 199 (0X00C7)

<u>Description</u>	(OS) DosDevIoctl Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDEVIOCTL
<u>Minor Code</u>	199 (0X00C7)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %W Category = %W Function = %W
	-----

OS2KRNL Major Code: 0X0005 Minor Code: 200 (0X00C8)

<u>Description</u>	(OS) DosDisConnectMmPipe Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDISCONNECTNMPIPE
<u>Minor Code</u>	200 (0X00C8)
<u>Trace Groups</u>	PIP
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %W
	-----

OS2KRNL Major Code: 0X0005 Minor Code: 201 (0X00C9)



<u><b>Description</b></u>	(OS) DosDupHandle Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSDUPHANDLE
<u><b>Minor Code</b></u>	201 (0X00C9)
<u><b>Trace Groups</b></u>	FS
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 202 (0X00CA)

<u><b>Description</b></u>	(OS) DosEditName Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSEEDITNAME
<u><b>Minor Code</b></u>	202 (0X00CA)
<u><b>Trace Groups</b></u>	FS
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Edit string = %S Source string = %S Level = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 203 (0X00CB)

<u><b>Description</b></u>	(OS) DosEnterCritSec Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSENTERCRITSEC
<u><b>Minor Code</b></u>	

203 (0X00CB)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

No parameters traced.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 204 (0X00CC)

**Description**

(OS) DosError Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSERROR

**Minor Code**

204 (0X00CC)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Error setting = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 206 (0X00CE)

**Description**

(OS) DosExit Dos Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSEXIT

**Minor Code**

206 (0X00CE)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Function = %W ResultCode = %W

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 207 (0X00CF)

Description	(OS) DosExitCritSec Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSEXITCRITSEC
Minor Code	207 (0X00CF)
Trace Groups	TK
Trace Types	PRE
Traced Parameters	No parameters traced.

# OS2KRNL Major Code: 0X0005 Minor Code: 208 (0X00D0)

Description	(OS) DosExitList Dos Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSEXITLIST
Minor Code	208 (0X00D0)
Trace Groups	TK
Trace Types	PRE
Traced Parameters	Function = %W   Address = %F

# OS2KRNL Major Code: 0X0005 Minor Code: 209 (0X00D1)

Description	(OS) DosFileIO Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFILEIO
Minor Code	209 (0X00D1)
Trace Groups	

FS  
Trace Types  
PRE

Traced Parameters  
  
Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 210 (0X00D2)

Description  
(OS) DosFileLocks Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFILELOCKS

Minor Code  
210 (0X00D2)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
  
Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 211 (0X00D3)

Description  
(OS) DosFindClose Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDCLOSE

Minor Code  
211 (0X00D3)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
  
Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 212 (0X00D4)

<u>Description</u>	(OS) DosFindFirst2 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDFIRST2
<u>Minor Code</u>	212 (0X00D4)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Path = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 213 (0X00D5)

<u>Description</u>	(OS) DosFindFirst Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDFIRST
<u>Minor Code</u>	213 (0X00D5)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Path = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 214 (0X00D6)

<u>Description</u>	(OS) DosFindFromName Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDFROMNAME
<u>Minor Code</u>	214 (0X00D6)
<u>Trace Groups</u>	FS

**Trace Types**  
PRE

**Traced Parameters**  
  
Name = %S  
Position = %F Directory Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 215 (0X00D7)

**Description**  
(OS) DosFindNext Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNEXT

**Minor Code**  
215 (0X00D7)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**  
  
Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 216 (0X00D8)

**Description**  
(OS) DosFindNotifyClose Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNOTIFYCLOSE

**Minor Code**  
216 (0X00D8)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**  
  
Handle = %W

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 217 (0X00D9)

Description	(OS) DosFindNotifyFirst Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNOTIFYFIRST
Minor Code	217 (0X00D9)
Trace Groups	FS
Trace Types	PRE
Traced Parameters	Name = %S Attribute = %W Level = %W Timeout = %W%W

# OS2KRNL Major Code: 0X0005 Minor Code: 218 (0X00DA)

Description	(OS) DosFindNotifyNext Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFINDNOTIFYNEXT
Minor Code	218 (0X00DA)
Trace Groups	FS
Trace Types	PRE
Traced Parameters	Change count = %W Handle = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 219 (0X00DB)

Description	(OS) DosFlagProcess Pre-Invocation
Tracepoint	

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFLAGPROCESS

**Minor Code**

219 (0X00DB)

**Trace Groups**

SIG

**Trace Types**

PRE

**Traced Parameters**

Pid = %W Action = %W

Signal = %W Arg = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 220 (0X00DC)

**Description**

(OS) DosFreeModule Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREEMODULE

**Minor Code**

220 (0X00DC)

**Trace Groups**

LDR

**Trace Types**

PRE

**Traced Parameters**

Module Handle = %F

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 221 (0X00DD)

**Description**

(OS) DosFreeSeg Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREESEG

**Minor Code**

221 (0X00DD)

**Trace Groups**

SEL

**Trace Types**

PRE

**Traced Parameters**



Selector = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 222 (0X00DE)

### Description

(OS) DosFSAttach Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFSATTACH

### Minor Code

222 (0X00DE)

### Trace Groups

FS

### Trace Types

PRE

### Traced Parameters

Device = %S

FSD = %S

OpFlag = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 223 (0X00DF)

### Description

(OS) DosFSCtl Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFSCTL

### Minor Code

223 (0X00DF)

### Trace Groups

FS

### Trace Types

PRE

### Traced Parameters

FSD = %S

Handle = %W Route = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 224 (0X00E0)

<b>Description</b>	(OS) DosGetDateTime Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETDATETIME
<b>Minor Code</b>	224 (0X00E0)
<b>Trace Groups</b>	TIM
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	DateTime structure at %W%W

## OS2KRNL Major Code: 0X0005 Minor Code: 225 (0X00E1)

<b>Description</b>	(OS) DosGetModHandle Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETMODHANDLE
<b>Minor Code</b>	225 (0X00E1)
<b>Trace Groups</b>	LDR
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	Module name = %S

## OS2KRNL Major Code: 0X0005 Minor Code: 226 (0X00E2)

<b>Description</b>	(OS) DosGetModName Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETMODNAME
<b>Minor Code</b>	226 (0X00E2)
<b>Trace Groups</b>	LDR

<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Module handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 227 (0X00E3)

<u>Description</u>	(OS) DosGetPid Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETPID
<u>Minor Code</u>	227 (0X00E3)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	No parameters traced.

## OS2KRNL Major Code: 0X0005 Minor Code: 228 (0X00E4)

<u>Description</u>	(OS) DosGetProcAddr Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETPROCADDR
<u>Minor Code</u>	228 (0X00E4)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Module handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 229 (0X00E5)

<u>Description</u>	(OS) DosGetPrty Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETPRTY
<u>Minor Code</u>	229 (0X00E5)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ID = %W Scope = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 230 (0X00E6)

<u>Description</u>	(OS) DosGetResource Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETRESOURCE
<u>Minor Code</u>	230 (0X00E6)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	NameID = %W Type = %W Module handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 231 (0X00E7)

<u>Description</u>	(OS) DosGetSeg Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETSEG
<u>Minor Code</u>	231 (0X00E7)
<u>Trace Groups</u>	

SEL

Trace Types

PRE

Traced Parameters

Selector = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 232 (0X00E8)

Description

(OS) DosGetShrSeg Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETSHRSEG

Minor Code

232 (0X00E8)

Trace Groups

SEL

Trace Types

PRE

Traced Parameters

Name = %S

-----

OS2KRNL Major Code: 0X0005 Minor Code: 233 (0X00E9)

Description

(OS) DosGetVersion Pre-Invocation

Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETVERSION

Minor Code

233 (0X00E9)

Trace Groups

TK

Trace Types

PRE

Traced Parameters

No parameters traced.

-----

OS2KRNL Major Code: 0X0005 Minor Code: 234 (0X00EA)

<u><b>Description</b></u>	(OS) DosGiveSeg Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGIVESEG
<u><b>Minor Code</b></u>	234 (0X00EA)
<u><b>Trace Groups</b></u>	SEL
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Selector = %W Pid = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 235 (0X00EB)

<u><b>Description</b></u>	(OS) DosHoldSignal Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSHOLDSIGNAL
<u><b>Minor Code</b></u>	235 (0X00EB)
<u><b>Trace Groups</b></u>	SIG
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Action = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 236 (0X00EC)

<u><b>Description</b></u>	(OS) DoslCopy Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSICOPY
<u><b>Minor Code</b></u>	236 (0X00EC)
<u><b>Trace Groups</b></u>	FS

**Trace Types**

PRE

**Traced Parameters**

Source = %S

Target = %S

OpMode = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 237 (0X00ED)

**Description**

(OS) DoslExecPgm Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIEXECPGM

**Minor Code**

237 (0X00ED)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Program name = %S

Exec flag = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 240 (0X00F0)

**Description**

(OS) DoslSetCP Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSISETCP

**Minor Code**

240 (0X00F0)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Codepage ID = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 241 (0X00F1)

<u>Description</u>	(OS) DosKillProcess Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSKILLPROCESS
<u>Minor Code</u>	241 (0X00F1)
<u>Trace Groups</u>	SIG
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	PID = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 242 (0X00F2)

<u>Description</u>	(OS) DosLoadModule Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSLOADMODULE
<u>Minor Code</u>	242 (0X00F2)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Module name = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 243 (0X00F3)

<u>Description</u>	(OS) DosMakeNmPipe Pre-Invocation
<u>Tracepoint</u>	



Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMAKENMPIPE

**Minor Code**

243 (0X00F3)

**Trace Groups**

PIP

**Trace Types**

PRE

**Traced Parameters**

Name = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 244 (0X00F4)

**Description**

(OS) DosMakePipe Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMAKEPIPE

**Minor Code**

244 (0X00F4)

**Trace Groups**

PIP

**Trace Types**

PRE

**Traced Parameters**

Size of pipe = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 245 (0X00F5)

**Description**

(OS) DosMkDir2 Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMKDIR2

**Minor Code**

245 (0X00F5)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Path = %S

---

## OS2KRNL Major Code: 0X0005 Minor Code: 246 (0X00F6)

### Description

(OS) DosMkDir Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMKDIR

### Minor Code

246 (0X00F6)

### Trace Groups

FS

### Trace Types

PRE

### Traced Parameters

Path = %S

---

## OS2KRNL Major Code: 0X0005 Minor Code: 252 (0X00FC)

### Description

(OS) DosMove Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSMOVE

### Minor Code

252 (0X00FC)

### Trace Groups

FS

### Trace Types

PRE

### Traced Parameters

New name = %S

Old name = %S

---

## OS2KRNL Major Code: 0X0005 Minor Code: 253 (0X00FD)

### Description

(OS) DosMuxSemWait Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.DOSMUXSEMWAIT

**Minor Code**

253 (0X00FD)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Timeout=%D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 254 (0X00FE)

**Description**

(OS) DosNewSize Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSNEWSIZE

**Minor Code**

254 (0X00FE)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Filesize %W%W Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 255 (0X00FF)

**Description**

(OS) DosOpen2 Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPEN2

**Minor Code**

255 (0X00FF)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Filename = %S  
Mode = %W Control = %W  
Attrib = %W Size = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 256 (0X0100)

**Description**

(OS) DosOpen Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPEN

**Minor Code**

256 (0X0100)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Filename = %S  
Mode = %W Control = %W  
Attrib = %W Size = %F

-----

OS2KRNL Major Code: 0X0005 Minor Code: 257 (0X0101)

**Description**

(OS) DosOpenSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPENSEM

**Minor Code**

257 (0X0101)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Name = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 258 (0X0102)

<u>Description</u>	(OS) DosOpLockRelease Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPLOCKRELEASE
<u>Minor Code</u>	258 (0X0102)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Kernel key = %W:%W  Flags = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 259 (0X0103)

<u>Description</u>	(OS) DosOpLockWait Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPLOCKWAIT
<u>Minor Code</u>	259 (0X0103)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	No parameters traced.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 260 (0X0104)

<u>Description</u>	(OS) DosPeekNmPipe Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPEEKNMPIPE
<u>Minor Code</u>	260 (0X0104)

Trace Groups  
PIP

Trace Types  
PRE

Traced Parameters  
  
Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 261 (0X0105)

Description  
(OS) DosPhysicalDisk Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPHYSICALDISK

Minor Code  
261 (0X0105)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
  
Function = %W Data Len = %W  
Parm Len = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 262 (0X0106)

Description  
(OS) DosQNmPipeState Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQNMPIPESEMSTATE

Minor Code  
262 (0X0106)

Trace Groups  
PIP

Trace Types  
PRE

Traced Parameters  
  
Semaphore handle = %W%W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 263 (0X0107)

<u>Description</u>	(OS) DosPortAccess Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPORTACCESS
<u>Minor Code</u>	263 (0X0107)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Last Port = %W First Port = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 265 (0X0109)

<u>Description</u>	(OS) DosQCurDir Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQCURDIR
<u>Minor Code</u>	265 (0X0109)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Drive Number = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 266 (0X010A)

<u>Description</u>	(OS) DosQCurDisk Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQCURDISK

**Minor Code** 266 (0X010A)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters** No parameters traced.

OS2KRNL Major Code: 0X0005 Minor Code: 267 (0X010B)

**Description** (OS) DosQFHandState Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFHANDSTATE

**Minor Code** 267 (0X010B)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters** File Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 268 (0X010C)

**Description** (OS) DosQFileInfo Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFILEINFO

**Minor Code** 268 (0X010C)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters** File Handle = %W Info Level = %W



# OS2KRNL Major Code: 0X0005 Minor Code: 269 (0X010D)

<u>Description</u>	(OS) DosQFileMode Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFILEMODE
<u>Minor Code</u>	269 (0X010D)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	File = %S

# OS2KRNL Major Code: 0X0005 Minor Code: 270 (0X010E)

<u>Description</u>	(OS) DosQFSAttach Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFSATTACH
<u>Minor Code</u>	270 (0X010E)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Path = %S Info Level = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 271 (0X010F)

<u>Description</u>	(OS) DosQFSInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQFSINFO

**Minor Code** 271 (0X010F)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**  
Drive = %W Info Level = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 272 (0X0110)

**Description** (OS) DosQHandType Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQHANDTYPE

**Minor Code** 272 (0X0110)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**  
Handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 273 (0X0111)

**Description** (OS) DosQNmPHandState Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQNMPHANDSTATE

**Minor Code** 273 (0X0111)

**Trace Groups** PIP

**Trace Types** PRE

**Traced Parameters**  
Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 274 (0X0112)

<u>Description</u>	(OS) DosQNmPipeInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQNMPIPEINFO
<u>Minor Code</u>	274 (0X0112)
<u>Trace Groups</u>	PIP
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 275 (0X0113)

<u>Description</u>	(OS) DosQPathInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQPATHINFO
<u>Minor Code</u>	275 (0X0113)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Path = %S
	Info Level = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 276 (0X0114)

<u>Description</u>	(OS) DosQSysInfo Pre-Invocation
--------------------	---------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQSYSINFO
<u>Minor Code</u>	276 (0X0114)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Index = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 277 (0X0115)

<u>Description</u>	(OS) DosQVerify Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSQVERIFY
<u>Minor Code</u>	277 (0X0115)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	No parameters traced.

## OS2KRNL Major Code: 0X0005 Minor Code: 278 (0X0116)

<u>Description</u>	(OS) DosRawReadNmPipe Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRAWREADNMPIPE
<u>Minor Code</u>	278 (0X0116)
<u>Trace Groups</u>	PIP
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

Handle = %W

---

## OS2KRNL Major Code: 0X0005 Minor Code: 279 (0X0117)

### Description

(OS) DosRawWriteNmPipe Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRAWWRITENMPIPE

### Minor Code

279 (0X0117)

### Trace Groups

PIP

### Trace Types

PRE

### Traced Parameters

Handle = %W

---

## OS2KRNL Major Code: 0X0005 Minor Code: 280 (0X0118)

### Description

(OS) DoslRead Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIREAD

### Minor Code

280 (0X0118)

### Trace Groups

FS

### Trace Types

PRE

### Traced Parameters

File Handle = %W Buffer = %W:%W

Buffer Size = %W Pointer to bytes read = %W:%W

---

## OS2KRNL Major Code: 0X0005 Minor Code: 282 (0X011A)

### Description

(OS) DosReallocHuge Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSREALLOCHUGE

**Minor Code**

282 (0X011A)

**Trace Groups**

SEL

**Trace Types**

PRE

**Traced Parameters**

New size = %W%W Selector = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 283 (0X011B)

**Description**

(OS) DosReallocSeg Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSREALLOCSEG

**Minor Code**

283 (0X011B)

**Trace Groups**

SEL

**Trace Types**

PRE

**Traced Parameters**

Selector = %W New size = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 284 (0X011C)

**Description**

(OS) DosResumeThread Dos Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRESUMETHREAD

**Minor Code**

284 (0X011C)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Function = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 285 (0X011D)

**Description**

(OS) DosRmdir Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRMDIR

**Minor Code**

285 (0X011D)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Directory = %S

-----

OS2KRNL Major Code: 0X0005 Minor Code: 286 (0X011E)

**Description**

(OS) DosSelectDisk Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSELECTDISK

**Minor Code**

286 (0X011E)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Drive Number = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 290 (0X0122)

<u>Description</u>	(OS) DosSemSetWait Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSEMSETWAIT
<u>Minor Code</u>	290 (0X0122)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Handle = %W%W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 292 (0X0124)

<u>Description</u>	(OS) DosSendSignal Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSEND SIGNAL
<u>Minor Code</u>	292 (0X0124)
<u>Trace Groups</u>	SIG
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Signal number = %W CSID = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 294 (0X0126)

<u>Description</u>	(OS) DosSetDateTime Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETDATETIME
<u>Minor Code</u>	294 (0X0126)
<u>Trace Groups</u>	TIM
<u>Trace Types</u>	PRE



**Traced Parameters**

DateTime structure at %W%W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 295 (0X0127)

**Description**

(OS) DosSetFHandState Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFHANDSTATE

**Minor Code**

295 (0X0127)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

File Handle = %W State = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 296 (0X0128)

**Description**

(OS) DosSetFileInfo Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFILEINFO

**Minor Code**

296 (0X0128)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Info Level = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 297 (0X0129)

<u>Description</u>	(OS) DosSetFileMode Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFILEMODE
<u>Minor Code</u>	297 (0X0129)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	File = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 298 (0X012A)

<u>Description</u>	(OS) DosSetFSInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETFSINFO
<u>Minor Code</u>	298 (0X012A)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Level = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 299 (0X012B)

<u>Description</u>	(OS) DosSetMaxFH Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETMAXFH
<u>Minor Code</u>	299 (0X012B)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE

**Traced Parameters**

Handles = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 300 (0X012C)

**Description**

(OS) DosSetNmPHandState Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETNMPHANDSTATE

**Minor Code**

300 (0X012C)

**Trace Groups**

PIP

**Trace Types**

PRE

**Traced Parameters**

Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 301 (0X012D)

**Description**

(OS) DosSetNmPipeSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETNMPIPESEM

**Minor Code**

301 (0X012D)

**Trace Groups**

PIP

**Trace Types**

PRE

**Traced Parameters**

Semaphore handle = %W%W Pipe handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 302 (0X012E)

<u>Description</u>	(OS) DosSetPathInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETPATHINFO
<u>Minor Code</u>	302 (0X012E)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Path = %S Info Level = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 303 (0X012F)

<u>Description</u>	(OS) DosSetPrty Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETPRTY
<u>Minor Code</u>	303 (0X012F)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Scope = %W Priority Class, Delta = %W ID = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 304 (0X0130)

<u>Description</u>	(OS) DosSetSigHandler Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETSIGHANDLER
<u>Minor Code</u>	304 (0X0130)
<u>Trace Groups</u>	

SIG

**Trace Types**

PRE

**Traced Parameters**

Signal number = %W Action = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 305 (0X0131)

**Description**

(OS) DosSetVec Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETVEC

**Minor Code**

305 (0X0131)

**Trace Groups**

SIG

**Trace Types**

PRE

**Traced Parameters**

Return address = %W%W Vector = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 306 (0X0132)

**Description**

(OS) DosSetVerify Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETVERIFY

**Minor Code**

306 (0X0132)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

No parameters traced.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 307 (0X0133)

<u>Description</u>	(OS) DosSleep Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSLEEP
<u>Minor Code</u>	307 (0X0133)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Timeout Interval = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 308 (0X0134)

<u>Description</u>	(OS) DosSuspendThread Dos Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSUSPENDTHREAD
<u>Minor Code</u>	308 (0X0134)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Function = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 309 (0X0135)

<u>Description</u>	(OS) DosSystemService Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSYSTEMSERVICE
<u>Minor Code</u>	309 (0X0135)
<u>Trace Groups</u>	TK

<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Function = %W
-----	

## OS2KRNL Major Code: 0X0005 Minor Code: 310 (0X0136)

<u>Description</u>	(OS) DosTimerAsync Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IASYNCTIMER
<u>Minor Code</u>	310 (0X0136)
<u>Trace Groups</u>	TIM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interval = %D
	Semaphore at %D
-----	

## OS2KRNL Major Code: 0X0005 Minor Code: 311 (0X0137)

<u>Description</u>	(OS) DosTimerStart Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ISTARTTIMER
<u>Minor Code</u>	311 (0X0137)
<u>Trace Groups</u>	TIM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Interval = %D
	Semaphore at %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 312 (0X0138)

<u>Description</u>	(OS) Dos32StopTimer Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32STOPTIMER
<u>Minor Code</u>	312 (0X0138)
<u>Trace Groups</u>	TIM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 313 (0X0139)

<u>Description</u>	(OS) DosTransactNmPipe Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSTRANSACTNMPIPE
<u>Minor Code</u>	313 (0X0139)
<u>Trace Groups</u>	PIP
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 314 (0X013A)

<u>Description</u>	(OS) DosWaitNmPipe Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSWAITNMPIPE



**Minor Code**  
314 (0X013A)

**Trace Groups**  
PIP

**Trace Types**  
PRE

**Traced Parameters**  
  
Name = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 315 (0X013B)

**Description**  
(OS) DosWrite Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSWRITE

**Minor Code**  
315 (0X013B)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**  
  
File Handle = %W Buffer = %W:%W  
Buffer Size = %W Pointer to bytes written = %W:%W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 323 (0X0143)

**Description**  
(OS) DosGetResource2 Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETRESOURCE2

**Minor Code**  
323 (0X0143)

**Trace Groups**  
LDR

**Trace Types**  
PRE

**Traced Parameters**

NameID = %W Type = %W

Module handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 324 (0X0144)

### Description

(OS) DosGetResource2 Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETRESOURCE2

### Minor Code

324 (0X0144)

### Trace Groups

LDR

### Trace Types

POST

### Traced Parameters

Address = %W:%W Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 325 (0X0145)

### Description

(OS) DosFreeResource Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREERESOURCE

### Minor Code

325 (0X0145)

### Trace Groups

LDR

### Trace Types

PRE

### Traced Parameters

Selector = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 326 (0X0146)

### Description

(OS) DosFreeResource Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREERESOURCE

**Minor Code**

326 (0X0146)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 327 (0X0147)

**Description**

(OS) Dos32GetResource Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32GETRESOURCE

**Minor Code**

327 (0X0147)

**Trace Groups**

LDR

**Trace Types**

PRE

**Traced Parameters**

hMod = %D TypeID = %D

NameID = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 328 (0X0148)

**Description**

(OS) Dos32GetResource Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32GETRESOURCE

**Minor Code**

328 (0X0148)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Address = %F Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 331 (0X014B)

**Description**

(OS) Dos32FreeRsource Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32FREERESOURCE

**Minor Code**

331 (0X014B)

**Trace Groups**

LDR

**Trace Types**

PRE

**Traced Parameters**

Pointer = %F

-----

OS2KRNL Major Code: 0X0005 Minor Code: 332 (0X014C)

**Description**

(OS) Dos32FreeResource Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32FREERESOURCE

**Minor Code**

332 (0X014C)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 333 (0X014D)

<u>Description</u>	(OS) Dos32QueryProcAddr Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYPROCADDR
<u>Minor Code</u>	333 (0X014D)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Name = %S
	hMod = %D Ord = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 334 (0X014E)

<u>Description</u>	(OS) Dos32QueryProcAddr Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYPROCADDR
<u>Minor Code</u>	334 (0X014E)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Proc Addr = %F Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 335 (0X014F)

<u>Description</u>	(OS) Dos32CreateEventSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATEEVENTSEM
<u>Minor Code</u>	335 (0X014F)
<u>Trace Groups</u>	

SEM

**Trace Types**

PRE

**Traced Parameters**

Name ptr = %F Attribs = %D

Initial State = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 336 (0X0150)

**Description**

(OS) Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATEEVENTSEM

**Minor Code**

336 (0X0150)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Handle = %D Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 337 (0X0151)

**Description**

(OS) Dos32OpenEventSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32OPENEVENTSEM

**Minor Code**

337 (0X0151)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Name Ptr = %F phev = %F

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 338 (0X0152)

<u>Description</u>	(OS) Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32OPENEVENTSEM
<u>Minor Code</u>	338 (0X0152)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Handle = %D Return code = %D

# OS2KRNL Major Code: 0X0005 Minor Code: 339 (0X0153)

<u>Description</u>	(OS) Dos32OpenEventSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CLOSEEVENTSEM
<u>Minor Code</u>	339 (0X0153)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Handle = %D

# OS2KRNL Major Code: 0X0005 Minor Code: 340 (0X0154)

<u>Description</u>	(OS) Dos32CloseEventSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CLOSEEVENTSEM
<u>Minor Code</u>	

340 (0X0154)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 341 (0X0155)

**Description**

(OS) Dos32ResetEventSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32RESETEVENTSEM

**Minor Code**

341 (0X0155)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Handle = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 342 (0X0156)

**Description**

(OS) Dos32ResetEventSem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32RESETEVENTSEM

**Minor Code**

342 (0X0156)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return code = %D



-----

## OS2KRNL Major Code: 0X0005 Minor Code: 343 (0X0157)

<u>Description</u>	(OS) Dos32PostEventSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32POSTEVENTSEM
<u>Minor Code</u>	343 (0X0157)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 344 (0X0158)

<u>Description</u>	(OS) Dos32PostEventSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32POSTEVENTSEM
<u>Minor Code</u>	344 (0X0158)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 345 (0X0159)

<u>Description</u>	(OS) Dos32WaitEvenSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32WAITEVENTSEM

**Minor Code** 345 (0X0159)

**Trace Groups** SEM

**Trace Types** PRE

**Traced Parameters**

Handle = %D Timeout = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 346 (0X015A)

**Description** (OS) Dos32WaitEventSem Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32WAITEVENTSEM

**Minor Code** 346 (0X015A)

**Trace Groups** SEM

**Trace Types** POST

**Traced Parameters**

Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 347 (0X015B)

**Description** (OS) Dos32QueryEventSem Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYEVENTSEM

**Minor Code** 347 (0X015B)

**Trace Groups** SEM

**Trace Types** PRE

**Traced Parameters**

Handle = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 348 (0X015C)

<u>Description</u>	(OS) Dos32QueryEventSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYEVENTSEM
<u>Minor Code</u>	348 (0X015C)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	State = %D Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 349 (0X015D)

<u>Description</u>	(OS) Dos32CreateMutexSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATEMUTEXSEM
<u>Minor Code</u>	349 (0X015D)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Name ptr = %F Attribs = %D Initial State = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 350 (0X015E)

<u>Description</u>	(OS) Dos32CreateMutexSem Post-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATEMUTEXSEM

**Minor Code**

350 (0X015E)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Handle = %D Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 351 (0X015F)

**Description**

(OS) Dos32OpenMutexSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32OPENMUTEXSEM

**Minor Code**

351 (0X015F)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Name ptr = %F Handle = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 352 (0X0160)

**Description**

(OS) Dos32OpenMutexSem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32OPENMUTEXSEM

**Minor Code**

352 (0X0160)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Handle = %D Return code = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 353 (0X0161)

**Description**

(OS) Dos32CloseMutexSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CLOSEMUTEXSEM

**Minor Code**

353 (0X0161)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Handle = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 354 (0X0162)

**Description**

(OS) Dos32CloseMutexSem Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CLOSEMUTEXSEM

**Minor Code**

354 (0X0162)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return code = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 355 (0X0163)

**Description**

(OS) Dos32RequestMutexSem Pre-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32REQUESTMUTEXSEM
<u>Minor Code</u>	355 (0X0163)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Handle = %D Timeout = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 356 (0X0164)

<u>Description</u>	(OS) Dos32RequestMutexSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32REQUESTMUTEXSEM
<u>Minor Code</u>	356 (0X0164)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 357 (0X0165)

<u>Description</u>	(OS) Dos32ReleaseMutexSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32RELEASEMUTEXSEM
<u>Minor Code</u>	357 (0X0165)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

Handle = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 358 (0X0166)

### Description

(OS) Dos32ReleaseMutexSem Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32RELEASEMUTEXSEM

### Minor Code

358 (0X0166)

### Trace Groups

SEM

### Trace Types

POST

### Traced Parameters

Return code = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 359 (0X0167)

### Description

(OS) Dos32QueryMutexSem Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYMUTEXSEM

### Minor Code

359 (0X0167)

### Trace Groups

SEM

### Trace Types

PRE

### Traced Parameters

Handle = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 360 (0X0168)

### Description

(OS) Dos32QueryMutexSem Post-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMUTEXSEM
<u>Minor Code</u>	360 (0X0168)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	PID Owner = %D TID Owner = %D Count = %D Return code = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 361 (0X0169)

<u>Description</u>	(OS) Dos32CreateMuxWaitSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CREATEMUXWAITSEM
<u>Minor Code</u>	361 (0X0169)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Name ptr = %F cSemRec = %D SemRec ptr = %F Attribs = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 362 (0X016A)

<u>Description</u>	(OS) Dos32CreateMuxWaitSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CREATEMUXWAITSEM
<u>Minor Code</u>	362 (0X016A)
<u>Trace Groups</u>	SEM



**Trace Types**  
POST

**Traced Parameters**  
  
Handle = %D Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 363 (0X016B)

**Description**  
(OS) Dos32OpenMuxWaitSem Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32OPENMUXWAITSEM

**Minor Code**  
363 (0X016B)

**Trace Groups**  
SEM

**Trace Types**  
PRE

**Traced Parameters**  
  
Name Ptr = %F phmux = %F

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 364 (0X016C)

**Description**  
(OS) Dos32OpenMuxWaitSem Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32OPENMUXWAITSEM

**Minor Code**  
364 (0X016C)

**Trace Groups**  
SEM

**Trace Types**  
POST

**Traced Parameters**  
  
Handle = %D Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 365 (0X016D)

<b>Description</b>	(OS) Dos32CloseMuxWaitSem Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CLOSEMUXWAITSEM
<b>Minor Code</b>	365 (0X016D)
<b>Trace Groups</b>	SEM
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	Handle = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 366 (0X016E)

<b>Description</b>	(OS) Dos32CloseMuxWaitSem Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CLOSEMUXWAITSEM
<b>Minor Code</b>	366 (0X016E)
<b>Trace Groups</b>	SEM
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Return code = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 367 (0X016F)

<b>Description</b>	(OS) Dos32WaitMuxWaitSem Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32WAITMUXWAITSEM
<b>Minor Code</b>	367 (0X016F)
<b>Trace Groups</b>	SEM

<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %D Timeout = %D
	-----

## OS2KRNL Major Code: 0X0005 Minor Code: 368 (0X0170)

<u>Description</u>	(OS) Dos32WaitMuxWaitSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32WAITMUXWAITSEM
<u>Minor Code</u>	368 (0X0170)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D
	-----

## OS2KRNL Major Code: 0X0005 Minor Code: 369 (0X0171)

<u>Description</u>	(OS) Dos32AddMuxWaitSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ADDMUXWAITSEM
<u>Minor Code</u>	369 (0X0171)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %D SemRec = %Q
	-----

## OS2KRNL Major Code: 0X0005 Minor Code: 370 (0X0172)

<u>Description</u>	(OS) Dos32AddMuxWaitSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ADDMUXWAITSEM
<u>Minor Code</u>	370 (0X0172)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 371 (0X0173)

<u>Description</u>	(OS) Dos32DeleteMuxWaitSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32DELETEMUXWAITSEM
<u>Minor Code</u>	371 (0X0173)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Mux Handle = %D Sem Handle = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 372 (0X0174)

<u>Description</u>	(OS) Dos32DeleteMuxWaitSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32DELETEMUXWAITSEM
<u>Minor Code</u>	372 (0X0174)
<u>Trace Groups</u>	SEM

Trace Types  
POST

Traced Parameters  
  
Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 373 (0X0175)

Description  
(OS) Dos32QueryMuxWaitSem Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYMUXWAITSEM

Minor Code  
373 (0X0175)

Trace Groups  
SEM

Trace Types  
PRE

Traced Parameters  
  
Handle = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 374 (0X0176)

Description  
(OS) Dos32QueryMuxWaitSem Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYMUXWAITSEM

Minor Code  
374 (0X0176)

Trace Groups  
SEM

Trace Types  
POST

Traced Parameters  
  
cSemRec = %D pSemRec = %F  
Attribs = %D Return code = %D

# OS2KRNL Major Code: 0X0005 Minor Code: 375 (0X0177)

<u>Description</u>	(OS) DosGetCP Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSGETCP
<u>Minor Code</u>	375 (0X0177)
<u>Trace Groups</u>	NLS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	No parameters traced.

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 376 (0X0178)

<u>Description</u>	(OS) DosGetCP Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSGETCP
<u>Minor Code</u>	376 (0X0178)
<u>Trace Groups</u>	NLS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	No parameters traced.

-----

# OS2KRNL Major Code: 0X0005 Minor Code: 377 (0X0179)

<u>Description</u>	(OS) Dos32AsyncTimer Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ASYNCTIMER
<u>Minor Code</u>	377 (0X0179)
<u>Trace Groups</u>	TIM

Trace Types  
PRE

Traced Parameters  
  
Interval = %D  
Semaphore at %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 378 (0X017A)

Description  
(OS) Dos32AsyncTimer Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32ASYNCTIMER

Minor Code  
378 (0X017A)

Trace Groups  
TIM

Trace Types  
POST

Traced Parameters  
  
Return code = %D Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 379 (0X017B)

Description  
(OS) Dos32StartTimer Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32STARTTIMER

Minor Code  
379 (0X017B)

Trace Groups  
TIM

Trace Types  
PRE

Traced Parameters  
  
Interval = %D  
Semaphore at %D

# OS2KRNL Major Code: 0X0005 Minor Code: 380 (0X017C)

<u>Description</u>	(OS) Dos32StartTimer Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32STARTTIMER
<u>Minor Code</u>	380 (0X017C)
<u>Trace Groups</u>	TIM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %D Handle = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 391 (0X0187)

<u>Description</u>	(OS) Dos32WaitThread Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32WAITTHREAD
<u>Minor Code</u>	391 (0X0187)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ThreadID = %D WaitOption = %D

# OS2KRNL Major Code: 0X0005 Minor Code: 392 (0X0188)

<u>Description</u>	(OS) Dos32WaitThread Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32WAITTHREAD
<u>Minor Code</u>	



392 (0X0188)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

ThreadID = %D Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 393 (0X0189)

**Description**

(OS) Cluster Allocate Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.Allocate

**Minor Code**

393 (0X0189)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Cluster Count Wanted = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 394 (0X018A)

**Description**

(OS) Cluster Allocate Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postAllocate

**Minor Code**

394 (0X018A)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Cluster Count Obtained = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 395 (0X018B)

<u>Description</u>	(OS) Cluster Release Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preRelease
<u>Minor Code</u>	395 (0X018B)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Cluster Release = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 397 (0X018D)

<u>Description</u>	(OS) File Lock/Unlock Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.w_LockOper
<u>Minor Code</u>	397 (0X018D)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Unlock Flag = %W Handle = %W Region Offset = %D Length = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 401 (0X0191)

<u>Description</u>	(OS) Thread Reschedule Post-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: OS2KRNL.postSchedNext

**Minor Code**

401 (0X0191)

**Trace Groups**

TK

**Trace Types**

POST

**Traced Parameters**

Switched to PID = %W TID = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 403 (0X0193)

**Description**

(OS) DosMuxSemWait Post-Invocation (No Wait)

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSMUXSEMWAIT2

**Minor Code**

403 (0X0193)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Index = %W Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 404 (0X0194)

**Description**

(OS) DosEnumAttribute Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSENUMATTRIBUTE

**Minor Code**

404 (0X0194)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

No parameters traced.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 405 (0X0195)

<u>Description</u>	(OS) DosEnumAttribute Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSENUMATTRIBUTE
<u>Minor Code</u>	405 (0X0195)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 406 (0X0196)

<u>Description</u>	(OS) DoslSetFileInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSISETFILEINFO
<u>Minor Code</u>	406 (0X0196)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Info Level = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 407 (0X0197)

<u>Description</u>	(OS) DoslSetFileInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSISETFILEINFO

**Minor Code** 407 (0X0197)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 408 (0X0198)

**Description** (OS) DoslSetPathInfo Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOSISETPATHINFO

**Minor Code** 408 (0X0198)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**

Path =%S

Info Level = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 409 (0X0199)

**Description** (OS) DoslSetPathInfo Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSISETPATHINFO

**Minor Code** 409 (0X0199)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 410 (0X019A)

### Description

(OS) Dos32QueryResource Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYRESOURCESIZE

### Minor Code

410 (0X019A)

### Trace Groups

LDR

### Trace Types

PRE

### Traced Parameters

hMod = %D TypeID = %D

NameID = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 411 (0X019B)

### Description

(OS) Dos32QueryResourceSize Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYRESOURCESIZE

### Minor Code

411 (0X019B)

### Trace Groups

LDR

### Trace Types

POST

### Traced Parameters

Address = %F Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 412 (0X019C)

### Description

(OS) DoslDevloctl Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIDEVIOCTL

**Minor Code**

412 (0X019C)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

No parameters traced.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 413 (0X019D)

**Description**

(OS) DoslDevloctl Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIDEVIOCTL

**Minor Code**

413 (0X019D)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 414 (0X019E)

**Description**

(OS) DoslSetRelMaxFH Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSISETRELMAXFH

**Minor Code**

414 (0X019E)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

No parameters traced.

-----

OS2KRNL Major Code: 0X0005 Minor Code: 415 (0X019F)

<u>Description</u>	(OS) DoslSetRelMaxFH Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSISETRELMAXFH
<u>Minor Code</u>	415 (0X019F)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 416 (0X01A0)

<u>Description</u>	(OS) Dos32InitializePorthole Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32INITIALIZEPORTHOLE
<u>Minor Code</u>	416 (0X01A0)
<u>Trace Groups</u>	LDR
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Init Routine = %D entry type = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 417 (0X01A1)

<u>Description</u>	(OS) Dos32InitializePorthole Post-Invocation
<u>Tracepoint</u>	



Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32INITIALIZEPORTHOLE

**Minor Code**

417 (0X01A1)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 418 (0X01A2)

**Description**

(OS) Dos32QueryHeaderInfo Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYHEADERINFO

**Minor Code**

418 (0X01A2)

**Trace Groups**

LDR

**Trace Types**

PRE

**Traced Parameters**

hMod = %D index = %D

phdr = %D cb = %D

subfunc = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 419 (0X01A3)

**Description**

(OS) Dos32QueryHeaderInfo Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYHEADERINFO

**Minor Code**

419 (0X01A3)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 420 (0X01A4)

**Description**

(OS) Dos32QueryProcType Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYPROCTYPE

**Minor Code**

420 (0X01A4)

**Trace Groups**

LDR

**Trace Types**

PRE

**Traced Parameters**

Name = %S

hMod = %D Ord = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 421 (0X01A5)

**Description**

(OS) Dos32QueryProcType Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYPROCTYPE

**Minor Code**

421 (0X01A5)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Proc Type = %F Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 424 (0X01A8)

<b>Description</b>	(OS) DosOpen2Compt Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPEN2COMPT
<b>Minor Code</b>	424 (0X01A8)
<b>Trace Groups</b>	FS
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	Filename = %S Mode = %W Control = %W Attrib = %W Size = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 425 (0X01A9)

<b>Description</b>	(OS) DosOpen2Compt Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPEN2COMPT
<b>Minor Code</b>	425 (0X01A9)
<b>Trace Groups</b>	FS
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Action = %W Handle = %W Return Code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 428 (0X01AC)

<b>Description</b>	(OS) Dos32ISetFHState Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32ISETFHSTATE

**Minor Code** 428 (0X01AC)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**

File Handle = %W State = %W %W

## OS2KRNL Major Code: 0X0005 Minor Code: 429 (0X01AD)

**Description** (OS) Dos32!SetFHState Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32!SETFHSTATE

**Minor Code** 429 (0X01AD)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return Code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 430 (0X01AE)

**Description** (OS) Dos32!QUERYFHSTATE Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32!QUERYFHSTATE

**Minor Code** 430 (0X01AE)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**

File Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 431 (0X01AF)

Description	(OS) Dos32IQUERYFHSTATE Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IQUERYFHSTATE
Minor Code	431 (0X01AF)
Trace Groups	FS
Trace Types	POST
Traced Parameters	Handle State = %W Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 432 (0X01B0)

Description	(OS) Dos32IRead Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IREAD
Minor Code	432 (0X01B0)
Trace Groups	FS
Trace Types	PRE
Traced Parameters	File Handle = %D Buffer = %F Buffer Size = %D Pointer to Bytes Read = %F

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 433 (0X01B1)

Description	(OS) Dos32IRead Post-Invocation
-------------	---------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IREAD
<u>Minor Code</u>	433 (0X01B1)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Bytes Read = %D Return code = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 434 (0X01B2)

<u>Description</u>	(OS) Dos32IWrite Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IWRITE
<u>Minor Code</u>	434 (0X01B2)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	File Handle = %D Buffer = %F
	Buffer Size = %D Pointer to Bytes Written = %F

## OS2KRNL Major Code: 0X0005 Minor Code: 435 (0X01B3)

<u>Description</u>	(OS) Dos32IWrite Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IWRITE
<u>Minor Code</u>	435 (0X01B3)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST

**Traced Parameters**

Bytes Written = %D Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 436 (0X01B4)

**Description**

(OS) Dos32DumpProcess Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32DUMPPROCESS

**Minor Code**

436 (0X01B4)

**Trace Groups**

TK

**Trace Types**

PRE

**Traced Parameters**

Flag = %D Drive = %D

PID = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 437 (0X01B5)

**Description**

(OS) Dos32DumpProcess Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32DUMPPROCESS

**Minor Code**

437 (0X01B5)

**Trace Groups**

TK

**Trace Types**

POST

**Traced Parameters**

Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 438 (0X01B6)

<u>Description</u>	(OS) Dos32SuppressPopUps Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SUPPRESSPOPUPS
<u>Minor Code</u>	438 (0X01B6)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Flag = %D Drive = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 439 (0X01B7)

<u>Description</u>	(OS) Dos32SuppressPopUps Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SUPPRESSPOPUPS
<u>Minor Code</u>	439 (0X01B7)
<u>Trace Groups</u>	TK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 440 (0X01B8)

<u>Description</u>	(OS) Dos32KillThread Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32KILLTHREAD
<u>Minor Code</u>	440 (0X01B8)
<u>Trace Groups</u>	TK



Trace Types  
PRE

Traced Parameters  
Tid = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 441 (0X01B9)

Description  
(OS) Dos32KillThread Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32KILLTHREAD

Minor Code  
441 (0X01B9)

Trace Groups  
TK

Trace Types  
POST

Traced Parameters  
Return code = %D

-----

OS2KRNL Major Code: 0X0005 Minor Code: 442 (0X01BA)

Description  
(OS) Dos32IProtectSetFHState Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IPROTECTSETFHSTATE

Minor Code  
442 (0X01BA)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
File Handle = %W State = %W %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 443 (0X01BB)

<u>Description</u>	(OS) Dos32IProtectSetFHState Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IPROTECTSETFHSTATE
<u>Minor Code</u>	443 (0X01BB)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 444 (0X01BC)

<u>Description</u>	(OS) Dos32IPROTECTQUERYFHSTATE Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IPROTECTQUERYFHSTATE
<u>Minor Code</u>	444 (0X01BC)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	File Handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 445 (0X01BD)

<u>Description</u>	(OS) Dos32IPROTECTQUERYFHSTATE Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IPROTECTQUERYFHSTATE
<u>Minor Code</u>	445 (0X01BD)
<u>Trace Groups</u>	FS

**Trace Types** POST

**Traced Parameters**  
Handle State = %W Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 446 (0X01BE)

**Description** (OS) DosProtectChgFilePtr Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTCHGFILEPTR

**Minor Code** 446 (0X01BE)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**  
Type = %W Distance = %W%W  
Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 447 (0X01BF)

**Description** (OS) DosProtectChgFilePtr Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTCHGFILEPTR

**Minor Code** 447 (0X01BF)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**  
Location = %W%W Return code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 448 (0X01C0)

Description	(OS) DosProtectClose Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTCLOSE
Minor Code	448 (0X01C0)
Trace Groups	FS
Trace Types	PRE
Traced Parameters	Handle = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 449 (0X01C1)

Description	(OS) DosProtectClose Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTCLOSE
Minor Code	449 (0X01C1)
Trace Groups	FS
Trace Types	POST
Traced Parameters	Return code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 450 (0X01C2)

Description	(OS) DosCloseChangeNotify Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCLOSECHANGENOTIFY
Minor Code	450 (0X01C2)

Trace Groups FS

Trace Types PRE

Traced Parameters

Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 451 (0X01C3)

Description (OS) DosCloseChangeNotify Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCLOSECHANGENOTIFY

Minor Code 451 (0X01C3)

Trace Groups FS

Trace Types POST

Traced Parameters

Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 452 (0X01C4)

Description (OS) DosProtectEnumAttribute Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTENUMATTRIBUTE

Minor Code 452 (0X01C4)

Trace Groups FS

Trace Types PRE

Traced Parameters No parameters traced.

-----

OS2KRNL Major Code: 0X0005 Minor Code: 453 (0X01C5)

<u>Description</u>	(OS) DosProtectEnumAttribute Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTENUMATTRIBUTE
<u>Minor Code</u>	453 (0X01C5)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 454 (0X01C6)

<u>Description</u>	(OS) DosProtectFileIO Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTFILEIO
<u>Minor Code</u>	454 (0X01C6)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 455 (0X01C7)

<u>Description</u>	(OS) DosProtectFileIO Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTFILEIO
<u>Minor Code</u>	455 (0X01C7)
<u>Trace Groups</u>	FS

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 456 (0X01C8)

Description  
(OS) DosProtectFileLocks Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTFILELOCKS

Minor Code  
456 (0X01C8)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
  
Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 457 (0X01C9)

Description  
(OS) DosProtectFileLocks Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTFILELOCKS

Minor Code  
457 (0X01C9)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
  
Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 458 (0X01CA)

<u>Description</u>	(OS) DosForceDelete Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFORCEDELETE
<u>Minor Code</u>	458 (0X01CA)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Path = %S

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 459 (0X01CB)

<u>Description</u>	(OS) DosForceDelete Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFORCEDELETE
<u>Minor Code</u>	459 (0X01CB)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 460 (0X01CC)

<u>Description</u>	(OS) DosIProtectRead Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIPROTECTREAD
<u>Minor Code</u>	460 (0X01CC)
<u>Trace Groups</u>	FS



Trace Types  
PRE

Traced Parameters  
Handle = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 461 (0X01CD)

Description  
(OS) DoslProtectRead Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIPROTECTREAD

Minor Code  
461 (0X01CD)

Trace Groups  
FS

Trace Types  
POST

Traced Parameters  
Bytes Read = %W Return code = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 462 (0X01CE)

Description  
(OS) DoslProtectSetFileInfo Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIPROTECTSETFILEINFO

Minor Code  
462 (0X01CE)

Trace Groups  
FS

Trace Types  
PRE

Traced Parameters  
Info Level = %W

-----

OS2KRNL Major Code: 0X0005 Minor Code: 463 (0X01CF)

<u>Description</u>	(OS) DoslProtectSetFileInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIPROTECTSETFILEINFO
<u>Minor Code</u>	463 (0X01CF)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 464 (0X01D0)

<u>Description</u>	(OS) DoslPROTECTWrite Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSIPROTECTWRITE
<u>Minor Code</u>	464 (0X01D0)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Handle = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 465 (0X01D1)

<u>Description</u>	(OS) DoslProtectWrite Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSIPROTECTWRITE
<u>Minor Code</u>	465 (0X01D1)
<u>Trace Groups</u>	FS

Trace Types POST

Traced Parameters

Bytes Written = %W Return code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 466 (0X01D2)

Description (OS) DosProtectNewSize Pre-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTNEWSIZE

Minor Code 466 (0X01D2)

Trace Groups FS

Trace Types PRE

Traced Parameters  
Filesize %W%W Handle = %W

OS2KRNL Major Code: 0X0005 Minor Code: 467 (0X01D3)

Description (OS) DosProtectNewSize Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTNEWSIZE

Minor Code 467 (0X01D3)

Trace Groups FS

Trace Types POST

Traced Parameters  
Return Code = %W

OS2KRNL Major Code: 0X0005 Minor Code: 468 (0X01D4)

<b>Description</b>	(OS) DOSPROTECTOPEN Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTOPEN
<b>Minor Code</b>	468 (0X01D4)
<b>Trace Groups</b>	FS
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	Filename = %S Mode = %W Control = %W Attrib = %W Size = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 469 (0X01D5)

<b>Description</b>	(OS) DosProtectOpen Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTOPEN
<b>Minor Code</b>	469 (0X01D5)
<b>Trace Groups</b>	FS
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Action = %W Handle = %W Return Code = %W

## OS2KRNL Major Code: 0X0005 Minor Code: 470 (0X01D6)

<b>Description</b>	(OS) DosOpenChangeNotify Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSOPENCHANGENOTIFY

<u>Minor Code</u>	470 (0X01D6)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	No parameters traced.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 471 (0X01D7)

<u>Description</u>	(OS) DosOpenChangeNotify Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSOPENCHANGENOTIFY
<u>Minor Code</u>	471 (0X01D7)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 472 (0X01D8)

<u>Description</u>	(OS) DosProtectQFHandState Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTQFHANDSTATE
<u>Minor Code</u>	472 (0X01D8)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	File Handle = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 473 (0X01D9)

<u>Description</u>	(OS) DosProtectQFHandState Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTQFHANDSTATE
<u>Minor Code</u>	473 (0X01D9)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Handle State = %W Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 474 (0X01DA)

<u>Description</u>	(OS) DosProtectQFileInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTQFILEINFO
<u>Minor Code</u>	474 (0X01DA)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	File Handle = %W Info Level = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 475 (0X01DB)

<u>Description</u>	(OS) DosProtectQFileInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTQFILEINFO

**Minor Code** 475 (0X01DB)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 476 (0X01DC)

**Description** (OS) DosResetChangeNotify Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRESETCHANGENOTIFY

**Minor Code** 476 (0X01DC)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters** No parameters traced.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 477 (0X01DD)

**Description** (OS) DosResetChangeNotify Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRESETCHANGENOTIFY

**Minor Code** 477 (0X01DD)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Return code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 478 (0X01DE)

Description	(OS) DosProtectSetFHandState Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTSETFHANDSTATE
Minor Code	478 (0X01DE)
Trace Groups	FS
Trace Types	PRE
Traced Parameters	File Handle = %W State = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 479 (0X01DF)

Description	(OS) DosProtectSetFHandState Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTSETFHANDSTATE
Minor Code	479 (0X01DF)
Trace Groups	FS
Trace Types	POST
Traced Parameters	Return Code = %W

# OS2KRNL Major Code: 0X0005 Minor Code: 480 (0X01E0)

Description	(OS) DosProtectSetFileInfo Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSPROTECTSETFILEINFO
Minor Code	



480 (0X01E0)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Info Level = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 481 (0X01E1)

**Description**

(OS) DosProtectSetFileInfo Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSPROTECTSETFILEINFO

**Minor Code**

481 (0X01E1)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 482 (0X01E2)

**Description**

(OS) Dos32PMPostEvenSem Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PMPOSTEVENTSEM

**Minor Code**

482 (0X01E2)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

PMHandle = %D Handle = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 483 (0X01E3)

<u>Description</u>	(OS) Dos32PMPostEventSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PMPOSTEVENTSEM
<u>Minor Code</u>	483 (0X01E3)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 484 (0X01E4)

<u>Description</u>	(OS) Dos32PMWaitEventSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PMWAITEVENTSEM
<u>Minor Code</u>	484 (0X01E4)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 485 (0X01E5)

<u>Description</u>	(OS) Dos32PMWaitMuxWaitSem Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PMWAITMUXWAITSEM

<u>Minor Code</u>	485 (0X01E5)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %D Handle = %D
	Timeout = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 486 (0X01E6)

<u>Description</u>	(OS) Dos32PMWaitMuxWaitSem Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PMWAITMUXWAITSEM
<u>Minor Code</u>	486 (0X01E6)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 487 (0X01E7)

<u>Description</u>	(OS) Dos32QueryExtLIBPATH Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32QUERYEXTLIBPATH
<u>Minor Code</u>	487 (0X01E7)
<u>Trace Groups</u>	VM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

Flags = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 488 (0X01E8)

### Description

(OS) Dos32QueryExtLIBPATH Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32QUERYEXTLIBPATH

### Minor Code

488 (0X01E8)

### Trace Groups

VM

### Trace Types

POST

### Traced Parameters

Path = %S Return code = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 489 (0X01E9)

### Description

(OS) Dos32SetExtLIBPATH Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SETEXTLIBPATH

### Minor Code

489 (0X01E9)

### Trace Groups

VM

### Trace Types

PRE

### Traced Parameters

Path = %S Flags = %D

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 490 (0X01EA)

### Description

(OS) Dos32SetExtLIBPATH Post-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SETEXTLIBPATH
<b><u>Minor Code</u></b>	490 (0X01EA)
<b><u>Trace Groups</u></b>	VM
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Return code = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 491 (0X01EB)

<b><u>Description</u></b>	(OS) Dos32VERIFYPIDTID Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32VERIFYPIDTID
<b><u>Minor Code</u></b>	491 (0X01EB)
<b><u>Trace Groups</u></b>	TK
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Pid = %D Tid = %D

## OS2KRNL Major Code: 0X0005 Minor Code: 492 (0X01EC)

<b><u>Description</u></b>	(OS) Dos32VERIFYPIDTID Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32VERIFYPIDTID
<b><u>Minor Code</u></b>	492 (0X01EC)
<b><u>Trace Groups</u></b>	TK
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	

Return code = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 493 (0X01ED)

### Description

(OS) Dos32PMWaitEvenSem Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PMWAITEVENTSEM

### Minor Code

493 (0X01ED)

### Trace Groups

SEM

### Trace Types

PRE

### Traced Parameters

PMHandle = %D Handle = %D

Timeout = %D

---

## OS2KRNL Major Code: 0X0005 Minor Code: 00494 (0X01EE)

### Description

(OS) Dos32CancelLockRequest Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32CANCELLOCKREQUEST

### Minor Code

494 (0X01EE)

### Trace Groups

FS

### Trace Types

PRE, API

### Traced Parameters

Offset = %F Range = %F Handle = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

---

## OS2KRNL Major Code: 0X0005 Minor Code: 00495 (0X01EF)

<u>Description</u>	(OS) Dos32CancelLockRequest Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32CANCELLOCKREQUEST
<u>Minor Code</u>	495 (0X01EF)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

## OS2KRNL Major Code: 0X0005 Minor Code: 00496 (0X01F0)

<u>Description</u>	(OS) Dos32SetFileLocks Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32SETFILELOCKS
<u>Minor Code</u>	496 (0X01F0)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	Handle = D% Unlock Offset = %F Range = %F" Lock Offset = %F Range = %F" Timeout = %D Flags = %D"

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

## OS2KRNL Major Code: 0X0005 Minor Code: 00497 (0X01F1)

<u>Description</u>	(OS) Dos32SetFileLocks Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32SETFILELOCKS
<u>Minor Code</u>	497 (0X01F1)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

## OS2KRNL Major Code: 0X0005 Minor Code: 00498 (0X01F2)

<u>Description</u>	(OS) Dos32ProtectSetFileLocks Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PROTECTSETFILELOCKS
<u>Minor Code</u>	498 (0X01F2)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	Handle = %D Unlock Offset = %F Range = %F" Lock Offset = %F Range = %F" Timeout = %D Flags = %D Lock Id = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

## OS2KRNL Major Code: 0X0005 Minor Code: 00499 (0X01F3)

<u>Description</u>	(OS) Dos32ProtectSetFileLocks Post-Invocation
--------------------	---



<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32PROTECTSETFILELOCKS
<u>Minor Code</u>	499 (0X01F3)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	
	Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

## OS2KRNL Major Code: 0X0005 Minor Code: 00500 (0X01F4)

<u>Description</u>	(OS) DosCreateSpinLock Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSCREATESPINLOCK
<u>Minor Code</u>	500 (0X01F4)
<u>Trace Groups</u>	LOCK
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	
	pLockHandle = %W:%W"

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

## OS2KRNL Major Code: 0X0005 Minor Code: 00501 (0X01F5)

<u>Description</u>	(OS) DosCreateSpinLock Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSCREATESPINLOCK
<u>Minor Code</u>	501 (0X01F5)
<u>Trace Groups</u>	

LOCK

**Trace Types**

POST, API

**Traced Parameters**

Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 00502 (0X01F6)

**Description**

(OS) DosAcquireSpinLock Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.preDOSACQUIRESPINLOCK

**Minor Code**

502 (0X01F6)

**Trace Groups**

LOCK

**Trace Types**

PRE, API

**Traced Parameters**

"Lock Handle = %W%W

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 00503 (0X01F7)

**Description**

(OS) DosAcquireSpinLock Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSACQUIRESPINLOCK

**Minor Code**

503 (0X01F7)

**Trace Groups**

LOCK

**Trace Types**

POST, API

**Traced Parameters**

No parameters traced.

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 00504 (0X01F8)

<u>Description</u>	(OS) DosReleaseSpinLock Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSRELEASESPINLOCK
<u>Minor Code</u>	504 (0X01F8)
<u>Trace Groups</u>	LOCK
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	

"Lock Handle = %W%W

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 00505 (0X01F9)

<u>Description</u>	(OS) DosReleaseSpinLock Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSRELEASESPINLOCK
<u>Minor Code</u>	505 (0X01F8)
<u>Trace Groups</u>	LOCK
<u>Trace Types</u>	POST, API
<u>Traced Parameters</u>	No parameters traced.

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 00506 (0X01FA)

<u>Description</u>	(OS) DosFreeSpinLock Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOSFREESPINLOCK
<u>Minor Code</u>	506 (0X01FA)
<u>Trace Groups</u>	LOCK
<u>Trace Types</u>	PRE, API
<u>Traced Parameters</u>	"Lock Handle = %W%W

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

## OS2KRNL Major Code: 0X0005 Minor Code: 00507 (0X01FB)

<u>Description</u>	(OS) DosFreeSpinLock Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.postDOSFREESPINLOCK
<u>Minor Code</u>	5071 (0X01FB)
<u>Trace Groups</u>	LOCK
<u>Trace Types</u>	POST, API
<u>Traced Parameters</u>	Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35.

## OS2KRNL Major Code: 0X0005 Minor Code: 00508 (0X01FC)

<u>Description</u>	(OS) Dos32IProtectRead Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32PROTECTSETFILELOCKS

**Minor Code** 508 (0X01FC)

**Trace Groups** FS

**Trace Types** PRE, API

**Traced Parameters**

File Handle = %D Buffer = %F

Buffer Size = %D Pointer to Bytes Read = %F

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

---

## OS2KRNL Major Code: 0X0005 Minor Code: 00509 (0X01FD)

**Description** (OS) Dos32IProtectRead Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IPROTECTREAD

**Minor Code** 509 (0X01FD)

**Trace Groups** FS

**Trace Types** POST, API

**Traced Parameters**

Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

---

## OS2KRNL Major Code: 0X0005 Minor Code: 00510 (0X01FE)

**Description** (OS) Dos32IProtectWrite Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2KRNL.preDOS32IPROTECTWRITE

**Minor Code** 510 (0X01FE)

**Trace Groups** FS

**Trace Types**  
PRE, API

**Traced Parameters**  
  
File Handle = %D Buffer = %F  
Buffer Size = %D Pointer to Bytes Written = %F

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 00511 (0X01FF)

**Description**  
(OS) Dos32IProtectWrite Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.postDOS32IPROTECTWRITE

**Minor Code**  
511 (0X01FF)

**Trace Groups**  
FS

**Trace Types**  
POST, API

**Traced Parameters**  
  
Return code = %D

**Note:** Tracepoint added with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 32768 (0X8000)

**Description**  
(OS) DosSetTraceInfo Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2KRNL.preDOSSETTRACEINFO

**Minor Code**  
32768 (0X8000)

**Trace Groups**  
TK

**Trace Types**  
PRE

**Traced Parameters**

Request Code = %W PID = %W

Count = %W

-----

## OS2KRNL Major Code: 0X0005 Minor Code: 65521 (0XFFF1)

### Description

(OS) DosSetTraceInfo Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2KRNL.postDOSSETTRACEINFO

### Minor Code

65521 (0XFFF1)

### Trace Groups

TK

### Trace Types

POST

### Traced Parameters

Return code = %W

-----

## DevHlp Services Trace Events

The tracepoints for the Device Helper Services major code are identified in the following tables. These tracepoints are static tracepoints. They are compiled with the code.

### **Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for DEVHELP Major Code: 0X0006, sorted by minor code.](#)

[Trace events for DEVHELP Major Code: 0X0006 ,sorted by tracepoint.](#)

-----

## Trace Events for DEVHELP Major Code: 0X0006, Sorted by Minor Code

00001 (0X0001) (OS) DevHlp\_ABIOSCall Pre-Invocation  
00002 (0X0002) (OS) DevHlp\_ABIOSCCommonEnt Pre-Invocation  
00003 (0X0003) (OS) DevHlp\_ABIOSGetParms Pre-Invocation  
00004 (0X0004) (OS) DevHlp\_AddTraceEvent Pre-Invocation  
00005 (0X0005) (OS) DevHlp\_AllocGDTSel Pre-Invocation  
00006 (0X0006) (OS) DevHlp\_AllocPhys Pre-Invocation  
00007 (0X0007) (OS) DevHlp\_AllocReqPacket Pre-Invocation  
00008 (0X0008) (OS) DevHlp\_AttachDD Pre-Invocation  
00009 (0X0009) (OS) DevHlp\_Block Pre-Invocation

00010 (0X000A) (OS) DevHlp\_DeRegister Pre-Invocation  
00011 (0X000B) (OS) DevHlp\_DevDone Pre-Invocation  
00012 (0X000C) (OS) DevHlp\_EOI Pre-Invocation  
00013 (0X000D) (OS) DevHlp\_FreeLIDEntry Pre-Invocation  
00014 (0X000E) (OS) DevHlp\_FreePhys Pre-Invocation  
00015 (0X000F) (OS) DevHlp\_FreeReqPacket Pre-Invocation  
00016 (0X0010) (OS) DevHlp\_GetDeviceBlock Pre-Invocation  
00017 (0X0011) (OS) DevHlp\_GetDOSVar Pre-Invocation  
00018 (0X0012) (OS) DevHlp\_GetLIDEntry Pre-Invocation  
00019 (0X0013) (OS) DevHlp\_InternalError Pre-Invocation  
00020 (0X0014) (OS) DevHlp\_Lock Pre-Invocation  
00021 (0X0015) (OS) DevHlp\_LogEntry Pre-Invocation  
00022 (0X0016) (OS) DevHlp\_MonCreate Pre-Invocation  
00023 (0X0017) (OS) DevHlp\_MonFlush Pre-Invocation  
00024 (0X0018) (OS) DevHlp\_MonWrite Pre-Invocation  
00025 (0X0019) (OS) DevHlp\_PhysToGDTSelector Pre-Invocation  
00026 (0X001A) (OS) DevHlp\_PhysToUVirt Pre-Invocation  
00027 (0X001B) (OS) DevHlp\_PhysToVirt Pre-Invocation  
00028 (0X001C) (OS) DevHlp\_Profiling Kernel Pre-Invocation  
00029 (0X001D) (OS) DevHlp\_ProtToReal Pre-Invocation  
00030 (0X001E) (OS) DevHlp\_PullParticular Pre-Invocation  
00031 (0X001F) (OS) DevHlp\_PullReqPacket Pre-Invocation  
00032 (0X0020) (OS) DevHlp\_PushReqPacket Pre-Invocation  
00033 (0X0021) (OS) DevHlp\_QueueFlush Pre-Invocation  
00034 (0X0022) (OS) DevHlp\_QueueInit Pre-Invocation  
00035 (0X0023) (OS) DevHlp\_QueueRead Pre-Invocation  
00036 (0X0024) (OS) DevHlp\_QueueWrite Pre-Invocation  
00037 (0X0025) (OS) DevHlp\_RealToProt Pre-Invocation  
00038 (0X0026) (OS) DevHlp\_Register Pre-Invocation  
00039 (0X0027) (OS) DevHlp\_RegStackUsage Pre-Invocation  
00040 (0X0028) (OS) DevHlp\_ResetTimer Pre-Invocation  
00042 (0X002A) (OS) DevHlp\_ROMCriticalSection Pre-Invocation  
00043 (0X002B) (OS) DevHlp\_ProcRun Pre-Invocation  
00044 (0X002C) (OS) DevHlp\_SchedClock Pre-Invocation  
00045 (0X002D) (OS) DevHlp\_SemClear Pre-Invocation  
00046 (0X002E) (OS) DevHlp\_SemHandle Pre-Invocation  
00047 (0X002F) (OS) DevHlp\_SemRequest Pre-Invocation  
00048 (0X0030) (OS) DevHlp\_SendEvent Pre-Invocation  
00049 (0X0031) (OS) DevHlp\_SetIRQ Pre-Invocation  
00050 (0X0032) (OS) DevHlp\_SetROMVector Pre-Invocation  
00051 (0X0033) (OS) DevHlp\_SetTimer Pre-Invocation  
00052 (0X0034) (OS) DevHlp\_SortReqPacket Pre-Invocation  
00053 (0X0035) (OS) DevHlp\_TCYield Pre-Invocation  
00054 (0X0036) (OS) DevHlp\_TickCount Pre-Invocation  
00055 (0X0037) (OS) DevHlp\_Unlock Pre-Invocation  
00056 (0X0038) (OS) DevHlp\_UnPhysToVirt Pre-Invocation  
00057 (0X0039) (OS) DevHlp\_UnSetIRQ Pre-Invocation  
00058 (0X003A) (OS) DevHlp\_VerifyAccess Pre-Invocation  
00060 (0X003C) (OS) DevHlp\_VirtToPhys Pre-Invocation  
00061 (0X003D) (OS) DevHlp\_VMAAlloc Pre-Invocation  
00062 (0X003E) (OS) DevHlp\_VMFree Pre-Invocation  
00063 (0X003F) (OS) DevHlp\_VMGlobalToProcess Pre-Invocation  
00065 (0X0041) (OS) DevHlp\_VMLock Pre-Invocation  
00066 (0X0042) (OS) DevHlp\_VMProcessToGlobal Pre-Invocation  
00067 (0X0043) (OS) DevHlp\_VMUnlock Pre-Invocation  
00069 (0X0045) (OS) DevHlp\_Yield Pre-Invocation  
00073 (0X0049) (OS) DevHlp\_FreeGDTSelector Pre-Invocation  
00074 (0X004A) (OS) DevHlp\_VirtToLin Pre-Invocation  
00075 (0X004B) (OS) DevHlp\_LinToGDTSelector Pre-Invocation  
00076 (0X004C) (OS) DevHlp\_GetDescInfo Pre-Invocation  
00077 (0X004D) (OS) DevHlp\_LinToPageList Pre-Invocation  
00078 (0X004E) (OS) DevHlp\_PageListToLin Pre-Invocation  
00079 (0X004F) (OS) DevHlp\_PageListToGDTSelector Pre-Invocation  
00081 (0X0051) (OS) DevHlp\_PhysToGDTSel Pre-Invocation  
00083 (0X0053) (OS) DevHlp\_AllocCtxHook Pre-Invocation  
00084 (0X0054) (OS) DevHlp\_ArmCtxHook Pre-Invocation  
00085 (0X0055) (OS) DevHlp\_FreeCtxHook Pre-Invocation  
00087 (0X0057) (OS) DevHlp\_OpenEventSem Pre-Invocation  
00088 (0X0058) (OS) DevHlp\_CloseEventSem Pre-Invocation  
00089 (0X0059) (OS) DevHlp\_PostEventSem Pre-Invocation  
00090 (0X005A) (OS) DevHlp\_ResetEventSem Pre-Invocation  
00093 (0X005D) (OS) DevHlp\_ProcRun2 Pre-Invocation  
00108 (0X006C) (OS) DevHlp\_ModifyPriority Pre-Invocation



00109 (0X006D) (OS) DevHlp\_RegisterTmrDD Pre-Invocation  
00110 (0X006E) (OS) DevHlp\_RegisterPerfCtrs Pre-Invocation  
00124 (0X007C) (OS) DevHlp\_RegisterPDD Pre-Invocation  
00125 (0X007D) (OS) DevHlp\_RegisterBeep Pre-Invocation  
00126 (0X007E) (OS) DevHlp\_Beep Pre-Invocation  
00129 (0X0081) (OS) DevHlp\_ABIOSCall Post-Invocation  
00130 (0X0082) (OS) DevHlp\_ABIOSCCommonEnt Post-Invocation  
00131 (0X0083) (OS) DevHlp\_ABIOSGetParms Post-Invocation  
00132 (0X0084) (OS) DevHlp\_AddTraceEvent Post-Invocation  
00133 (0X0085) (OS) DevHlp\_AllocGDTSel Post-Invocation  
00134 (0X0086) (OS) DevHlp\_AllocPhys Post-Invocation  
00135 (0X0087) (OS) DevHlp\_AllocReqPacket Post-Invocation  
00136 (0X0088) (OS) DevHlp\_AttachDD Post-Invocation  
00137 (0X0089) (OS) DevHlp\_Block Post-Invocation  
00138 (0X008A) (OS) DevHlp\_DeRegister Post-Invocation  
00139 (0X008B) (OS) DevHlp\_DevDone Post-Invocation  
00140 (0X008C) (OS) DevHlp\_EOI Post-Invocation  
00141 (0X008D) (OS) DevHlp\_FreeLIDEntry Post-Invocation  
00142 (0X008E) (OS) DevHlp\_FreePhys Post-Invocation  
00143 (0X008F) (OS) DevHlp\_FreeReqPacket Post-Invocation  
00144 (0X0090) (OS) DevHlp\_GetDeviceBlock Post-Invocation  
00145 (0X0091) (OS) DevHlp\_GetDOSVar Post-Invocation  
00146 (0X0092) (OS) DevHlp\_GetLIDEntry Post-Invocation  
00147 (0X0093) (OS) DevHlp\_InternalError Post-Invocation  
00148 (0X0094) (OS) DevHlp\_Lock Post-Invocation  
00149 (0X0095) (OS) DevHlp\_LogEntry Post-Invocation  
00150 (0X0096) (OS) DevHlp\_MonCreate Post-Invocation  
00151 (0X0097) (OS) DevHlp\_MonFlush Post-Invocation  
00152 (0X0098) (OS) DevHlp\_MonWrite Post-Invocation  
00153 (0X0099) (OS) DevHlp\_PhysToGDTSelector Post-Invocation  
00154 (0X009A) (OS) DevHlp\_PhysToUVirt Post-Invocation  
00155 (0X009B) (OS) DevHlp\_PhysToVirt Post-Invocation  
00156 (0X009C) (OS) DevHlp\_Profiling Kernel Post-Invocation  
00157 (0X009D) (OS) DevHlp\_ProtToReal Post-Invocation  
00158 (0X009E) (OS) DevHlp\_PullParticular Post-Invocation  
00159 (0X009F) (OS) DevHlp\_PullReqPacket Post-Invocation  
00160 (0X00A0) (OS) DevHlp\_PushReqPacket Post-Invocation  
00161 (0X00A1) (OS) DevHlp\_QueueFlush Post-Invocation  
00162 (0X00A2) (OS) DevHlp\_QueueInit Post-Invocation  
00163 (0X00A3) (OS) DevHlp\_QueueRead Post-Invocation  
00164 (0X00A4) (OS) DevHlp\_QueueWrite Post-Invocation  
00165 (0X00A5) (OS) DevHlp\_RealToProt Post-Invocation  
00166 (0X00A6) (OS) DevHlp\_Register Post-Invocation  
00167 (0X00A7) (OS) DevHlp\_RegStackUsage Post-Invocation  
00168 (0X00A8) (OS) DevHlp\_ResetTimer Post-Invocation  
00170 (0X00AA) (OS) DevHlp\_ROMCritSection Post-Invocation  
00171 (0X00AB) (OS) DevHlp\_ProcRun Post-Invocation  
00172 (0X00AC) (OS) DevHlp\_SchedClock Post-Invocation  
00173 (0X00AD) (OS) DevHlp\_SemClear Post-Invocation  
00174 (0X00AE) (OS) DevHlp\_SemHandle Post-Invocation  
00175 (0X00AF) (OS) DevHlp\_SemRequest Post-Invocation  
00176 (0X00B0) (OS) DevHlp\_SendEvent Post-Invocation  
00177 (0X00B1) (OS) DevHlp\_SetIRQ Post-Invocation  
00178 (0X00B2) (OS) DevHlp\_SetROMVector Post-Invocation  
00179 (0X00B3) (OS) DevHlp\_SetTimer Post-Invocation  
00180 (0X00B4) (OS) DevHlp\_SortReqPacket Post-Invocation  
00181 (0X00B5) (OS) DevHlp\_TCYield Post-Invocation  
00182 (0X00B6) (OS) DevHlp\_TickCount Post-Invocation  
00183 (0X00B7) (OS) DevHlp\_Unlock Post-Invocation  
00184 (0X00B8) (OS) DevHlp\_UnPhysToVirt Post-Invocation  
00185 (0X00B9) (OS) DevHlp\_UnSetIRQ Post-Invocation  
00186 (0X00BA) (OS) DevHlp\_VerifyAccess Post-Invocation  
00188 (0X00BC) (OS) DevHlp\_VirtToPhys Post-Invocation  
00189 (0X00BD) (OS) DevHlp\_VMAlloc Post-Invocation  
00190 (0X00BE) (OS) DevHlp\_VMFree Post-Invocation  
00191 (0X00BF) (OS) DevHlp\_VMGlobalToProcess Post-Invocation  
00193 (0X00C1) (OS) DevHlp\_VMLock Post-Invocation  
00194 (0X00C2) (OS) DevHlp\_VMProcessToGlobal Post-Invocation  
00195 (0X00C3) (OS) DevHlp\_VMUnlock Post-Invocation  
00197 (0X00C5) (OS) DevHlp\_Yield Post-Invocation  
00201 (0X00C9) (OS) DevHlp\_FreeGDTSelector Post-Invocation  
00202 (0X00CA) (OS) DevHlp\_VirtToLin Post-Invocation  
00203 (0X00CB) (OS) DevHlp\_LinToGDTSelector Post-Invocation

00204 (0X00CC) (OS) DevHlp\_GetDescInfo Post-Invocation  
00205 (0X00CD) (OS) DevHlp\_LinToPageList Post-Invocation  
00206 (0X00CE) (OS) DevHlp\_PageListToLin Post-Invocation  
00207 (0X00CF) (OS) DevHlp\_PageListToGDTSelector Post-Invocation  
00209 (0X00D1) (OS) DevHlp\_PhysToGDTSel Post-Invocation  
00211 (0X00D3) (OS) DevHlp\_AllocCtxHook Post-Invocation  
00212 (0X00D4) (OS) DevHlp\_ArmCtxHook Post-Invocation  
00213 (0X00D5) (OS) DevHlp\_FreeCtxHook Post-Invocation  
00215 (0X00D7) (OS) DevHlp\_OpenEventSem Post-Invocation  
00216 (0X00D8) (OS) DevHlp\_CloseEventSem Post-Invocation  
00217 (0X00D9) (OS) DevHlp\_PostEventSem Post-Invocation  
00218 (0X00DA) (OS) DevHlp\_ResetEventSem Post-Invocation  
00221 (0X00DD) (OS) DevHlp\_ProcRun2 Post-Invocation  
00236 (0X00EC) (OS) DevHlp\_ModifyPriority Post-Invocation  
00237 (0X00ED) (OS) DevHlp\_RegisterTmrDD Post-Invocation  
00238 (0X00EE) (OS) DevHlp\_RegisterPerfCtrs Post-Invocation  
00252 (0X00FC) (OS) DevHlp\_RegisterPDD Post-Invocation  
00253 (0X00FD) (OS) DevHlp\_RegisterBeep Post-Invocation  
00254 (0X00FE) (OS) DevHlp\_Beep Post-Invocation

---

## Trace Events for DEVHELP Major Code: 0X0006, Sorted by Tracepoint

(OS) DevHlp\_ABIOSCall Post-Invocation 00129 (0X0081)  
(OS) DevHlp\_ABIOSCall Pre-Invocation 00001 (0X0001)  
(OS) DevHlp\_ABIOSCCommonEnt Post-Invocation 00130 (0X0082)  
(OS) DevHlp\_ABIOSCCommonEnt Pre-Invocation 00002 (0X0002)  
(OS) DevHlp\_ABIOSGetParms Post-Invocation 00131 (0X0083)  
(OS) DevHlp\_ABIOSGetParms Pre-Invocation 00003 (0X0003)  
(OS) DevHlp\_AddTraceEvent Post-Invocation 00132 (0X0084)  
(OS) DevHlp\_AddTraceEvent Pre-Invocation 00004 (0X0004)  
(OS) DevHlp\_AllocCtxHook Post-Invocation 00211 (0X00D3)  
(OS) DevHlp\_AllocCtxHook Pre-Invocation 00083 (0X0053)  
(OS) DevHlp\_AllocGDTSel Post-Invocation 00133 (0X0085)  
(OS) DevHlp\_AllocGDTSel Pre-Invocation 00005 (0X0005)  
(OS) DevHlp\_AllocPhys Post-Invocation 00134 (0X0086)  
(OS) DevHlp\_AllocPhys Pre-Invocation 00006 (0X0006)  
(OS) DevHlp\_AllocReqPacket Post-Invocation 00135 (0X0087)  
(OS) DevHlp\_AllocReqPacket Pre-Invocation 00007 (0X0007)  
(OS) DevHlp\_ArmCtxHook Post-Invocation 00212 (0X00D4)  
(OS) DevHlp\_ArmCtxHook Pre-Invocation 00084 (0X0054)  
(OS) DevHlp\_AttachDD Post-Invocation 00136 (0X0088)  
(OS) DevHlp\_AttachDD Pre-Invocation 00008 (0X0008)  
(OS) DevHlp\_Beep Post-Invocation 00254 (0X00FE)  
(OS) DevHlp\_Beep Pre-Invocation 00126 (0X007E)  
(OS) DevHlp\_Block Post-Invocation 00137 (0X0089)  
(OS) DevHlp\_Block Pre-Invocation 00009 (0X0009)  
(OS) DevHlp\_CloseEventSem Post-Invocation 00216 (0X00D8)  
(OS) DevHlp\_CloseEventSem Pre-Invocation 00088 (0X0058)  
(OS) DevHlp\_DeRegister Post-Invocation 00138 (0X008A)  
(OS) DevHlp\_DeRegister Pre-Invocation 00010 (0X000A)  
(OS) DevHlp\_DevDone Post-Invocation 00139 (0X008B)  
(OS) DevHlp\_DevDone Pre-Invocation 00011 (0X000B)  
(OS) DevHlp\_EOI Post-Invocation 00140 (0X008C)  
(OS) DevHlp\_EOI Pre-Invocation 00012 (0X000C)  
(OS) DevHlp\_FreeCtxHook Post-Invocation 00213 (0X00D5)  
(OS) DevHlp\_FreeCtxHook Pre-Invocation 00085 (0X0055)  
(OS) DevHlp\_FreeGDTSelector Post-Invocation 00201 (0X00C9)  
(OS) DevHlp\_FreeGDTSelector Pre-Invocation 00073 (0X0049)  
(OS) DevHlp\_FreeLIDEntry Post-Invocation 00141 (0X008D)  
(OS) DevHlp\_FreeLIDEntry Pre-Invocation 00013 (0X000D)  
(OS) DevHlp\_FreePhys Post-Invocation 00142 (0X008E)  
(OS) DevHlp\_FreePhys Pre-Invocation 00014 (0X000E)  
(OS) DevHlp\_FreeReqPacket Post-Invocation 00143 (0X008F)

(OS) DevHlp\_FreeReqPacket Pre-Invocation 00015 (0X000F)  
(OS) DevHlp\_GetDOSVar Post-Invocation 00145 (0X0091)  
(OS) DevHlp\_GetDOSVar Pre-Invocation 00017 (0X0011)  
(OS) DevHlp\_GetDescInfo Post-Invocation 00204 (0X00CC)  
(OS) DevHlp\_GetDescInfo Pre-Invocation 00076 (0X004C)  
(OS) DevHlp\_GetDeviceBlock Post-Invocation 00144 (0X0090)  
(OS) DevHlp\_GetDeviceBlock Pre-Invocation 00016 (0X0010)  
(OS) DevHlp\_GetLIDEntry Post-Invocation 00146 (0X0092)  
(OS) DevHlp\_GetLIDEntry Pre-Invocation 00018 (0X0012)  
(OS) DevHlp\_InternalError Post-Invocation 00147 (0X0093)  
(OS) DevHlp\_InternalError Pre-Invocation 00019 (0X0013)  
(OS) DevHlp\_LinToGDTSelector Post-Invocation 00203 (0X00CB)  
(OS) DevHlp\_LinToGDTSelector Pre-Invocation 00075 (0X004B)  
(OS) DevHlp\_LinToPageList Post-Invocation 00205 (0X00CD)  
(OS) DevHlp\_LinToPageList Pre-Invocation 00077 (0X004D)  
(OS) DevHlp\_Lock Post-Invocation 00148 (0X0094)  
(OS) DevHlp\_Lock Pre-Invocation 00020 (0X0014)  
(OS) DevHlp\_LogEntry Post-Invocation 00149 (0X0095)  
(OS) DevHlp\_LogEntry Pre-Invocation 00021 (0X0015)  
(OS) DevHlp\_ModifyPriority Post-Invocation 00236 (0X00EC)  
(OS) DevHlp\_ModifyPriority Pre-Invocation 00108 (0X006C)  
(OS) DevHlp\_MonCreate Post-Invocation 00150 (0X0096)  
(OS) DevHlp\_MonCreate Pre-Invocation 00022 (0X0016)  
(OS) DevHlp\_MonFlush Post-Invocation 00151 (0X0097)  
(OS) DevHlp\_MonFlush Pre-Invocation 00023 (0X0017)  
(OS) DevHlp\_MonWrite Post-Invocation 00152 (0X0098)  
(OS) DevHlp\_MonWrite Pre-Invocation 00024 (0X0018)  
(OS) DevHlp\_OpenEventSem Post-Invocation 00215 (0X00D7)  
(OS) DevHlp\_OpenEventSem Pre-Invocation 00087 (0X0057)  
(OS) DevHlp\_PageListToGDTSelector Post-Invocation 00207 (0X00CF)  
(OS) DevHlp\_PageListToGDTSelector Pre-Invocation 00079 (0X004F)  
(OS) DevHlp\_PageListToLin Post-Invocation 00206 (0X00CE)  
(OS) DevHlp\_PageListToLin Pre-Invocation 00078 (0X004E)  
(OS) DevHlp\_PhysToGDTSel Post-Invocation 00209 (0X00D1)  
(OS) DevHlp\_PhysToGDTSel Pre-Invocation 00081 (0X0051)  
(OS) DevHlp\_PhysToGDTSelector Post-Invocation 00153 (0X0099)  
(OS) DevHlp\_PhysToGDTSelector Pre-Invocation 00025 (0X0019)  
(OS) DevHlp\_PhysToUVirt Post-Invocation 00154 (0X009A)  
(OS) DevHlp\_PhysToUVirt Pre-Invocation 00026 (0X001A)  
(OS) DevHlp\_PhysToVirt Post-Invocation 00155 (0X009B)  
(OS) DevHlp\_PhysToVirt Pre-Invocation 00027 (0X001B)  
(OS) DevHlp\_PostEventSem Post-Invocation 00217 (0X00D9)  
(OS) DevHlp\_PostEventSem Pre-Invocation 00089 (0X0059)  
(OS) DevHlp\_ProcRun Post-Invocation 00171 (0X00AB)  
(OS) DevHlp\_ProcRun Pre-Invocation 00043 (0X002B)  
(OS) DevHlp\_ProcRun2 Post-Invocation 00221 (0X00DD)  
(OS) DevHlp\_ProcRun2 Pre-Invocation 00093 (0X005D)  
(OS) DevHlp\_Profiling Kernel Post-Invocation 00156 (0X009C)  
(OS) DevHlp\_Profiling Kernel Pre-Invocation 00028 (0X001C)  
(OS) DevHlp\_ProtToReal Post-Invocation 00157 (0X009D)  
(OS) DevHlp\_ProtToReal Pre-Invocation 00029 (0X001D)  
(OS) DevHlp\_PullParticular Post-Invocation 00158 (0X009E)  
(OS) DevHlp\_PullParticular Pre-Invocation 00030 (0X001E)  
(OS) DevHlp\_PullReqPacket Post-Invocation 00159 (0X009F)  
(OS) DevHlp\_PullReqPacket Pre-Invocation 00031 (0X001F)  
(OS) DevHlp\_PushReqPacket Post-Invocation 00160 (0X00A0)  
(OS) DevHlp\_PushReqPacket Pre-Invocation 00032 (0X0020)  
(OS) DevHlp\_QueueFlush Post-Invocation 00161 (0X00A1)  
(OS) DevHlp\_QueueFlush Pre-Invocation 00033 (0X0021)  
(OS) DevHlp\_QueueInit Post-Invocation 00162 (0X00A2)  
(OS) DevHlp\_QueueInit Pre-Invocation 00034 (0X0022)  
(OS) DevHlp\_QueueRead Post-Invocation 00163 (0X00A3)  
(OS) DevHlp\_QueueRead Pre-Invocation 00035 (0X0023)  
(OS) DevHlp\_QueueWrite Post-Invocation 00164 (0X00A4)  
(OS) DevHlp\_QueueWrite Pre-Invocation 00036 (0X0024)  
(OS) DevHlp\_ROMCriticalSection Post-Invocation 00170 (0X00AA)  
(OS) DevHlp\_ROMCriticalSection Pre-Invocation 00042 (0X002A)  
(OS) DevHlp\_RealToProt Post-Invocation 00165 (0X00A5)  
(OS) DevHlp\_RealToProt Pre-Invocation 00037 (0X0025)  
(OS) DevHlp\_RegStackUsage Post-Invocation 00167 (0X00A7)  
(OS) DevHlp\_RegStackUsage Pre-Invocation 00039 (0X0027)  
(OS) DevHlp\_Register Post-Invocation 00166 (0X00A6)  
(OS) DevHlp\_Register Pre-Invocation 00038 (0X0026)

(OS) DevHlp\_RegisterBeep Post-Invocation 00253 (0X00FD)  
 (OS) DevHlp\_RegisterBeep Pre-Invocation 00125 (0X007D)  
 (OS) DevHlp\_RegisterPDD Post-Invocation 00252 (0X00FC)  
 (OS) DevHlp\_RegisterPDD Pre-Invocation 00124 (0X007C)  
 (OS) DevHlp\_RegisterPerfCtrs Post-Invocation 00238 (0X00EE)  
 (OS) DevHlp\_RegisterPerfCtrs Pre-Invocation 00110 (0X006E)  
 (OS) DevHlp\_RegisterTmrDD Post-Invocation 00237 (0X00ED)  
 (OS) DevHlp\_RegisterTmrDD Pre-Invocation 00109 (0X006D)  
 (OS) DevHlp\_ResetEventSem Post-Invocation 00218 (0X00DA)  
 (OS) DevHlp\_ResetEventSem Pre-Invocation 00090 (0X005A)  
 (OS) DevHlp\_ResetTimer Post-Invocation 00168 (0X00A8)  
 (OS) DevHlp\_ResetTimer Pre-Invocation 00040 (0X0028)  
 (OS) DevHlp\_SchedClock Post-Invocation 00172 (0X00AC)  
 (OS) DevHlp\_SchedClock Pre-Invocation 00044 (0X002C)  
 (OS) DevHlp\_SemClear Post-Invocation 00173 (0X00AD)  
 (OS) DevHlp\_SemClear Pre-Invocation 00045 (0X002D)  
 (OS) DevHlp\_SemHandle Post-Invocation 00174 (0X00AE)  
 (OS) DevHlp\_SemHandle Pre-Invocation 00046 (0X002E)  
 (OS) DevHlp\_SemRequest Post-Invocation 00175 (0X00AF)  
 (OS) DevHlp\_SemRequest Pre-Invocation 00047 (0X002F)  
 (OS) DevHlp\_SendEvent Post-Invocation 00176 (0X00B0)  
 (OS) DevHlp\_SendEvent Pre-Invocation 00048 (0X0030)  
 (OS) DevHlp\_SetIRQ Post-Invocation 00177 (0X00B1)  
 (OS) DevHlp\_SetIRQ Pre-Invocation 00049 (0X0031)  
 (OS) DevHlp\_SetROMVector Post-Invocation 00178 (0X00B2)  
 (OS) DevHlp\_SetROMVector Pre-Invocation 00050 (0X0032)  
 (OS) DevHlp\_SetTimer Post-Invocation 00179 (0X00B3)  
 (OS) DevHlp\_SetTimer Pre-Invocation 00051 (0X0033)  
 (OS) DevHlp\_SortReqPacket Post-Invocation 00180 (0X00B4)  
 (OS) DevHlp\_SortReqPacket Pre-Invocation 00052 (0X0034)  
 (OS) DevHlp\_TCYield Post-Invocation 00181 (0X00B5)  
 (OS) DevHlp\_TCYield Pre-Invocation 00053 (0X0035)  
 (OS) DevHlp\_TickCount Post-Invocation 00182 (0X00B6)  
 (OS) DevHlp\_TickCount Pre-Invocation 00054 (0X0036)  
 (OS) DevHlp\_UnPhysToVirt Post-Invocation 00184 (0X00B8)  
 (OS) DevHlp\_UnPhysToVirt Pre-Invocation 00056 (0X0038)  
 (OS) DevHlp\_UnSetIRQ Post-Invocation 00185 (0X00B9)  
 (OS) DevHlp\_UnSetIRQ Pre-Invocation 00057 (0X0039)  
 (OS) DevHlp\_Unlock Post-Invocation 00183 (0X00B7)  
 (OS) DevHlp\_Unlock Pre-Invocation 00055 (0X0037)  
 (OS) DevHlp\_VMAlloc Post-Invocation 00189 (0X00BD)  
 (OS) DevHlp\_VMAlloc Pre-Invocation 00061 (0X003D)  
 (OS) DevHlp\_VMFree Post-Invocation 00190 (0X00BE)  
 (OS) DevHlp\_VMFree Pre-Invocation 00062 (0X003E)  
 (OS) DevHlp\_VMGlobalToProcess Post-Invocation 00191 (0X00BF)  
 (OS) DevHlp\_VMGlobalToProcess Pre-Invocation 00063 (0X003F)  
 (OS) DevHlp\_VMLock Post-Invocation 00193 (0X00C1)  
 (OS) DevHlp\_VMLock Pre-Invocation 00065 (0X0041)  
 (OS) DevHlp\_VMProcessToGlobal Post-Invocation 00194 (0X00C2)  
 (OS) DevHlp\_VMProcessToGlobal Pre-Invocation 00066 (0X0042)  
 (OS) DevHlp\_VMUnlock Post-Invocation 00195 (0X00C3)  
 (OS) DevHlp\_VMUnlock Pre-Invocation 00067 (0X0043)  
 (OS) DevHlp\_VerifyAccess Post-Invocation 00186 (0X00BA)  
 (OS) DevHlp\_VerifyAccess Pre-Invocation 00058 (0X003A)  
 (OS) DevHlp\_VirtToLin Post-Invocation 00202 (0X00CA)  
 (OS) DevHlp\_VirtToLin Pre-Invocation 00074 (0X004A)  
 (OS) DevHlp\_VirtToPhys Post-Invocation 00188 (0X00BC)  
 (OS) DevHlp\_VirtToPhys Pre-Invocation 00060 (0X003C)  
 (OS) DevHlp\_Yield Post-Invocation 00197 (0X00C5)  
 (OS) DevHlp\_Yield Pre-Invocation 00069 (0X0045)

-----

## DEVHELP Major Code: 0X0006 Minor Code: 1 (0X0001)

### Description

(OS) DevHlp\_ABIOSCall Pre-Invocation

<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	1 (0X0001)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	LID=%W Entry_Point=%W ABIOS Request Block=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 2 (0X0002)

<u>Description</u>	(OS) DevHlp_ABIOSTCommonEnt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	2 (0X0002)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Entry_Point=%W ABIOS Request Block=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 3 (0X0003)

<u>Description</u>	(OS) DevHlp_ABIOSTGetParms Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	3 (0X0003)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

LID=%W Parm\_Addr=%A

---

## DEVHELP Major Code: 0X0006 Minor Code: 4 (0X0004)

<u>Description</u>	(OS) DevHlp_AddTraceEvent Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	4 (0X0004)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

---

## DEVHELP Major Code: 0X0006 Minor Code: 5 (0X0005)

<u>Description</u>	(OS) DevHlp_AllocGDTSel Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	5 (0X0005)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	pSelectorArray=%A Number of GDT Selectors=%W

---

## DEVHELP Major Code: 0X0006 Minor Code: 6 (0X0006)

<u>Description</u>	(OS) DevHlp_AllocPhys Pre-Invocation
--------------------	--------------------------------------

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 6 (0X0006)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

size\_low=%W size\_high=%W relative position=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 7 (0X0007)

**Description** (OS) DevHlp\_AllocReqPacket Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 7 (0X0007)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters** No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 8 (0X0008)

**Description** (OS) DevHlp\_AttachDD Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 8 (0X0008)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters** No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 9 (0X0009)

<u>Description</u>	(OS) DevHlp_Block Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	9 (0X0009)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	event_id_low=%W event_id_high=%W timeout interval=%D interruptible_flag=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 10 (0X000A)

<u>Description</u>	(OS) DevHlp_DeRegister Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	10 (0X000A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	monitor_PID=%W monitor_handle=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 11 (0X000B)

<u>Description</u>	(OS) DevHlp_DevDone Pre-Invocation
<u>Tracepoint</u>	



Static tracepoint in DEVHELP.

**Minor Code**

11 (0X000B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 12 (0X000C)

**Description**

(OS) DevHlp\_EOI Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

12 (0X000C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

IRQNum=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 13 (0X000D)

**Description**

(OS) DevHlp\_FreeLIDEntry Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

13 (0X000D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

LID=%W Device Driver DS=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 14 (0X000E)

<u>Description</u>	(OS) DevHlp_FreePhys Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	14 (0X000E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	address_low=%W address_high=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 15 (0X000F)

<u>Description</u>	(OS) DevHlp_FreeReqPacket Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	15 (0X000F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	request_packet=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 16 (0X0010)

<u>Description</u>	(OS) DevHlp_GetDeviceBlock Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.

**Minor Code** 16 (0X0010)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Logical ID=%W Device Driver Data Seg=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 17 (0X0011)

**Description** (OS) DevHlp\_GetDOSVar Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 17 (0X0011)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

variable index=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 18 (0X0012)

**Description** (OS) DevHlp\_GetLIDEntry Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 18 (0X0012)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

DeviceId=%W RelativeID#=%W

Device Driver DS=%W Device State=%W

---

## DEVHELP Major Code: 0X0006 Minor Code: 19 (0X0013)

<u>Description</u>	(OS) DevHlp_InternalError Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	19 (0X0013)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

---

## DEVHELP Major Code: 0X0006 Minor Code: 20 (0X0014)

<u>Description</u>	(OS) DevHlp_Lock Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	20 (0X0014)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	segment=%W duration=%B block flag=%B

---

## DEVHELP Major Code: 0X0006 Minor Code: 21 (0X0015)

<u>Description</u>	(OS) DevHlp_LogEntry Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.

<u>Minor Code</u>	21 (0X0015)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 22 (0X0016)

<u>Description</u>	(OS) DevHlp_MonCreate Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	22 (0X0016)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	final_buffer=%A notify_rtn=%A Handle=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 23 (0X0017)

<u>Description</u>	(OS) DevHlp_MonFlush Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	23 (0X0017)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	monitor_handle=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 24 (0X0018)

<u>Description</u>	(OS) DevHlp_MonWrite Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	24 (0X0018)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	data_record=%A count=%W monitor_handle=%W wait_flag=%W timeout_high=%W timeout_low=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 25 (0X0019)

<u>Description</u>	(OS) DevHlp_PhysToGDTSelector Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	25 (0X0019)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Address=%D Size=%W Selector=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 26 (0X001A)

<u>Description</u>	(OS) DevHlp_PhysToUVirt Pre-Invocation
<u>Tracepoint</u>	

Static tracepoint in DEVHELP.

**Minor Code**

26 (0X001A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

address\_low=%W address\_high=%W

segment size=%W function code=%W

---

## DEVHELP Major Code: 0X0006 Minor Code: 27 (0X001B)

**Description**

(OS) DevHlp\_PhysToVirt Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

27 (0X001B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

address to convert=%D

---

## DEVHELP Major Code: 0X0006 Minor Code: 28 (0X001C)

**Description**

(OS) DevHlp\_Profiling Kernel Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

28 (0X001C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 29 (0X001D)

**Description**

(OS) DevHlp\_ProtToReal Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

29 (0X001D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Device Header=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 30 (0X001E)

**Description**

(OS) DevHlp\_PullParticular Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

30 (0X001E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Queue=%A request\_packet=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 31 (0X001F)

**Description**

(OS) DevHlp\_PullReqPacket Pre-Invocation



<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	31 (0X001F)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Queue=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 32 (0X0020)

<b><u>Description</u></b>	(OS) DevHlp_PushReqPacket Pre-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	32 (0X0020)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Queue=%A request_packet=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 33 (0X0021)

<b><u>Description</u></b>	(OS) DevHlp_QueueFlush Pre-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	33 (0X0021)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

Queue=%A

---

## DEVHELP Major Code: 0X0006 Minor Code: 34 (0X0022)

**Description** (OS) DevHlp\_QueueInit Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 34 (0X0022)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
Queue=%A

---

## DEVHELP Major Code: 0X0006 Minor Code: 35 (0X0023)

**Description** (OS) DevHlp\_QueueRead Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 35 (0X0023)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
Queue=%A

---

## DEVHELP Major Code: 0X0006 Minor Code: 36 (0X0024)

**Description** (OS) DevHlp\_QueueWrite Pre-Invocation

<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	36 (0X0024)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Queue=%A Character written=%B

-----

## DEVHELP Major Code: 0X0006 Minor Code: 37 (0X0025)

<u>Description</u>	(OS) DevHlp_RealToProt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	37 (0X0025)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Device Header=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 38 (0X0026)

<u>Description</u>	(OS) DevHlp_Register Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	38 (0X0026)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

monitor segment=%W input\_buffer=%W output\_buffer\_offset=%W  
monitor\_PID=%W monitor\_handle=%W placement\_flag=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 39 (0X0027)

<u>Description</u>	(OS) DevHlp_RegStackUsage Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	39 (0X0027)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 40 (0X0028)

<u>Description</u>	(OS) DevHlp_ResetTimer Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	40 (0X0028)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 42 (0X002A)

<u>Description</u>	(OS) DevHlp_ROMCriticalSection Pre-Invocation
--------------------	---

<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	42 (0X002A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	enter_exit flag=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 43 (0X002B)

<b><u>Description</u></b>	(OS) DevHlp_ProcRun Pre-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	43 (0X002B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	event_id_low=%W event_id_high=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 44 (0X002C)

<b><u>Description</u></b>	(OS) DevHlp_SchedClock Pre-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	44 (0X002C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 45 (0X002D)

<u>Description</u>	(OS) DevHlp_SemClear Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	45 (0X002D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	sem_handle_low=%W sem_handle_high=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 46 (0X002E)

<u>Description</u>	(OS) DevHlp_SemHandle Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	46 (0X002E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	sem_key_low=%W sem_key_high=%W usage flag=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 47 (0X002F)

<u>Description</u>	(OS) DevHlp_SemRequest Pre-Invocation
<u>Tracepoint</u>	

Static tracepoint in DEVHELP.

**Minor Code**

47 (0X002F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

sem\_handle\_low=%W sem\_handle\_high=%W timeout value=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 48 (0X0030)

**Description**

(OS) DevHlp\_SendEvent Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

48 (0X0030)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Event=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 49 (0X0031)

**Description**

(OS) DevHlp\_SetIRQ Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

49 (0X0031)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

IRQnum=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 50 (0X0032)

<u>Description</u>	(OS) DevHlp_SetROMVector Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	50 (0X0032)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 51 (0X0033)

<u>Description</u>	(OS) DevHlp_SetTimer Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	51 (0X0033)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 52 (0X0034)

<u>Description</u>	(OS) DevHlp_SortReqPacket Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	



52 (0X0034)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Queue=%A request\_packet=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 53 (0X0035)

**Description**

(OS) DevHlp\_TCYield Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

53 (0X0035)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 54 (0X0036)

**Description**

(OS) DevHlp\_TickCount Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

54 (0X0036)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

tick counts=%W

-----

# DEVHELP Major Code: 0X0006 Minor Code: 55 (0X0037)

<u>Description</u>	(OS) DevHlp_Unlock Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	55 (0X0037)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	lock_handle_low=%W lock_handle_high=%W

# DEVHELP Major Code: 0X0006 Minor Code: 56 (0X0038)

<u>Description</u>	(OS) DevHlp_UnPhysToVirt Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	56 (0X0038)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

# DEVHELP Major Code: 0X0006 Minor Code: 57 (0X0039)

<u>Description</u>	(OS) DevHlp_UnSetIRQ Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	57 (0X0039)
<u>Trace Groups</u>	

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

IRQnum=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 58 (0X003A)

**Description**

(OS) DevHlp\_VerifyAccess Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

58 (0X003A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Target Selector/Segment=%W Length=%W

Offset=%W                      Type of Access=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 60 (0X003C)

**Description**

(OS) DevHlp\_VirtToPhys Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

60 (0X003C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

address=%D

-----

# DEVHELP Major Code: 0X0006 Minor Code: 61 (0X003D)

<u>Description</u>	(OS) DevHlp_VMAlloc Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	61 (0X003D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Flags=%D Size=%D PhysAddr=%F

# DEVHELP Major Code: 0X0006 Minor Code: 62 (0X003E)

<u>Description</u>	(OS) DevHlp_VMFree Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	62 (0X003E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Linear address=%F

# DEVHELP Major Code: 0X0006 Minor Code: 63 (0X003F)

<u>Description</u>	(OS) DevHlp_VMGlobalToProcess Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	

63 (0X003F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Action flags=%D Linear address=%F Length=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 65 (0X0041)

**Description**

(OS) DevHlp\_VMLock Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

65 (0X0041)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Action flags=%D Linear address=%F Length=%D

pPageList=%F pLockHandle=%F

-----

## DEVHELP Major Code: 0X0006 Minor Code: 66 (0X0042)

**Description**

(OS) DevHlp\_VMProcessToGlobal Pre-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

66 (0X0042)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Action flags=%D Linear address=%F Length=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 67 (0X0043)

<u>Description</u>	(OS) DevHlp_VMUnlock Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	67 (0X0043)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	LockHandle=%d%d%d

-----

## DEVHELP Major Code: 0X0006 Minor Code: 69 (0X0045)

<u>Description</u>	(OS) DevHlp_Yield Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	69 (0X0045)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 73 (0X0049)

<u>Description</u>	(OS) DevHlp_FreeGDTSelector Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.

**Minor Code** 73 (0X0049)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Selector=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 74 (0X004A)

**Description** (OS) DevHlp\_VirtToLin Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 74 (0X004A)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Selector=%W Offset=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 75 (0X004B)

**Description** (OS) DevHlp\_LinToGDTSelector Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 75 (0X004B)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Selector=%W Linear Address=%F Size=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 76 (0X004C)

<u>Description</u>	(OS) DevHlp_GetDescInfo Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	76 (0X004C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Selector=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 77 (0X004D)

<u>Description</u>	(OS) DevHlp_LinToPageList Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	77 (0X004D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Linear Address=%F Size=%D pPageList=%F

-----

## DEVHELP Major Code: 0X0006 Minor Code: 78 (0X004E)

<u>Description</u>	(OS) DevHlp_PageListToLin Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.



**Minor Code** 78 (0X004E)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Size=%D pPageList=%F

## -----

### DEVHELP Major Code: 0X0006 Minor Code: 79 (0X004F)

**Description** (OS) DevHlp\_PageListToGDTSelector Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 79 (0X004F)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Selector=%W Size=%D Access=%D pPageList=%F

## -----

### DEVHELP Major Code: 0X0006 Minor Code: 81 (0X0051)

**Description** (OS) DevHlp\_PhysToGDTSel Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 81 (0X0051)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Address=%D Size=%D Access=%W Selector=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 83 (0X0053)

<u>Description</u>	(OS) DevHlp_AllocCtxHook Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	83 (0X0053)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Hook Handler=%D Context=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 84 (0X0054)

<u>Description</u>	(OS) DevHlp_ArmCtxHook Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	84 (0X0054)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Hook Data=%D Hook Handle=%D Context=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 85 (0X0055)

<u>Description</u>	(OS) DevHlp_FreeCtxHook Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.

**Minor Code** 85 (0X0055)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Hook Handle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 87 (0X0057)

**Description** (OS) DevHlp\_OpenEventSem Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 87 (0X0057)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Sem Handle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 88 (0X0058)

**Description** (OS) DevHlp\_CloseEventSem Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 88 (0X0058)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Sem Handle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 89 (0X0059)

<u>Description</u>	(OS) DevHlp_PostEventSem Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	89 (0X0059)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Sem Handle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 90 (0X005A)

<u>Description</u>	(OS) DevHlp_ResetEventSem Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	90 (0X005A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Sem Handle=%D, Post Count=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 93 (0X005D)

<u>Description</u>	(OS) DevHlp_ProcRun2 Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.

**Minor Code** 93 (0X005D)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

event\_id\_low=%W event\_id\_high=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 108 (0X006C)

**Description** (OS) DevHlp\_ModifyPriority Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 108 (0X006C)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Thread number=%W Keyboard switch=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 109 (0X006D)

**Description** (OS) DevHlp\_RegisterTmrDD Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 109 (0X006D)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

PTTimer0 entry point=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 110 (0X006E)

<u>Description</u>	(OS) DevHlp_RegisterPerfCtrs Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	110 (0X006E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	pCounterBlock=%A pTextBlock=%A Flags=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 124 (0X007C)

<u>Description</u>	(OS) DevHlp_RegisterPDD Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	124 (0X007C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Entry point=%A PDD name=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 125 (0X007D)

<u>Description</u>	(OS) DevHlp_RegisterBeep Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.

**Minor Code** 125 (0X007D)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

PTDBeep entry point=%A

## -----

## DEVHELP Major Code: 0X0006 Minor Code: 126 (0X007E)

**Description** (OS) DevHlp\_Beep Pre-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 126 (0X007E)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Beep frequency=%W Beep duration=%W

## -----

## DEVHELP Major Code: 0X0006 Minor Code: 129 (0X0081)

**Description** (OS) DevHlp\_ABIOSCall Post-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 129 (0X0081)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 130 (0X0082)

<u>Description</u>	(OS) DevHlp_ABIOSCommonEnt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	130 (0X0082)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 131 (0X0083)

<u>Description</u>	(OS) DevHlp_ABIOSGetParms Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	131 (0X0083)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 132 (0X0084)

<u>Description</u>	(OS) DevHlp_AddTraceEvent Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.



<b><u>Minor Code</u></b>	132 (0X0084)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 133 (0X0085)

<b><u>Description</u></b>	(OS) DevHlp_AllocGDTSel Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	133 (0X0085)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 134 (0X0086)

<b><u>Description</u></b>	(OS) DevHlp_AllocPhys Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	134 (0X0086)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return Code=%W Physical Address=%D

# DEVHELP Major Code: 0X0006 Minor Code: 135 (0X0087)

<u>Description</u>	(OS) DevHlp_AllocReqPacket Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	135 (0X0087)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	return code=%W   request packet=%A

# DEVHELP Major Code: 0X0006 Minor Code: 136 (0X0088)

<u>Description</u>	(OS) DevHlp_AttachDD Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	136 (0X0088)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Return code=%W

# DEVHELP Major Code: 0X0006 Minor Code: 137 (0X0089)

<u>Description</u>	(OS) DevHlp_Block Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	

137 (0X0089)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 138 (0X008A)

**Description**

(OS) DevHlp\_DeRegister Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

138 (0X008A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W Number of monitors=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 139 (0X008B)

**Description**

(OS) DevHlp\_DevDone Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

139 (0X008B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 140 (0X008C)

<b><u>Description</u></b>	(OS) DevHlp_EOI Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	140 (0X008C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 141 (0X008D)

<b><u>Description</u></b>	(OS) DevHlp_FreeLIDEntry Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	141 (0X008D)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 142 (0X008E)

<b><u>Description</u></b>	(OS) DevHlp_FreePhys Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	142 (0X008E)
<b><u>Trace Groups</u></b>	No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 143 (0X008F)

Description  
(OS) DevHlp\_FreeReqPacket Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
143 (0X008F)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 144 (0X0090)

Description  
(OS) DevHlp\_GetDeviceBlock Post-Invocation

Tracepoint  
Static tracepoint in DEVHELP.

Minor Code  
144 (0X0090)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
Return Code=%W Real Mode Pointer=%A Protect Mode Pointer=%A

-----

DEVHELP Major Code: 0X0006 Minor Code: 145 (0X0091)

<u>Description</u>	(OS) DevHlp_GetDOSVar Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	145 (0X0091)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Return code=%W
-----	

DEVHELP Major Code: 0X0006 Minor Code: 146 (0X0092)

<u>Description</u>	(OS) DevHlp_GetLIDEntry Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	146 (0X0092)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Return code=%W LID=%W
-----	

DEVHELP Major Code: 0X0006 Minor Code: 147 (0X0093)

<u>Description</u>	(OS) DevHlp_InternalError Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	147 (0X0093)
<u>Trace Groups</u>	No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters** No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 148 (0X0094)

**Description** (OS) DevHlp\_Lock Post-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 148 (0X0094)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

return code=%W lock handle=%D

-----

DEVHELP Major Code: 0X0006 Minor Code: 149 (0X0095)

**Description** (OS) DevHlp\_LogEntry Post-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 149 (0X0095)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters** No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 150 (0X0096)

**Description**

(OS) DevHlp\_MonCreate Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

150 (0X0096)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W    Handle=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 151 (0X0097)

**Description**

(OS) DevHlp\_MonFlush Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

151 (0X0097)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 152 (0X0098)

**Description**

(OS) DevHlp\_MonWrite Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

152 (0X0098)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.



**Traced Parameters**

Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 153 (0X0099)

**Description**

(OS) DevHlp\_PhysToGDTSelector Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

153 (0X0099)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 154 (0X009A)

**Description**

(OS) DevHlp\_PhysToUVirt Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

154 (0X009A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

return code=%W selector=%W offset=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 155 (0X009B)

<b><u>Description</u></b>	(OS) DevHlp_PhysToVirt Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	155 (0X009B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 156 (0X009C)

<b><u>Description</u></b>	(OS) DevHlp_Profiling Kernel Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	156 (0X009C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 157 (0X009D)

<b><u>Description</u></b>	(OS) DevHlp_ProtToReal Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	157 (0X009D)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 158 (0X009E)

### Description

(OS) DevHlp\_PullParticular Post-Invocation

### Tracepoint

Static tracepoint in DEVHELP.

### Minor Code

158 (0X009E)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 159 (0X009F)

### Description

(OS) DevHlp\_PullReqPacket Post-Invocation

### Tracepoint

Static tracepoint in DEVHELP.

### Minor Code

159 (0X009F)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

return code=%W request packet=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 160 (0X00A0)

### Description

(OS) DevHlp\_PushReqPacket Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

160 (0X00A0)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 161 (0X00A1)

**Description**

(OS) DevHlp\_QueueFlush Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

161 (0X00A1)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 162 (0X00A2)

**Description**

(OS) DevHlp\_QueueInit Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

162 (0X00A2)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 163 (0X00A3)

<u>Description</u>	(OS) DevHlp_QueueRead Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	163 (0X00A3)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Return code=%W Character read=%B

-----

## DEVHELP Major Code: 0X0006 Minor Code: 164 (0X00A4)

<u>Description</u>	(OS) DevHlp_QueueWrite Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	164 (0X00A4)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 165 (0X00A5)

<u>Description</u>	(OS) DevHlp_RealToProt Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.

**Minor Code** 165 (0X00A5)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 166 (0X00A6)

**Description** (OS) DevHlp\_Register Post-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 166 (0X00A6)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 167 (0X00A7)

**Description** (OS) DevHlp\_RegStackUsage Post-Invocation

**Tracepoint** Static tracepoint in DEVHELP.

**Minor Code** 167 (0X00A7)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 168 (0X00A8)

<u>Description</u>	(OS) DevHlp_ResetTimer Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	168 (0X00A8)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 170 (0X00AA)

<u>Description</u>	(OS) DevHlp_ROMCriticalSection Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	170 (0X00AA)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 171 (0X00AB)

<u>Description</u>	(OS) DevHlp_ProcRun Post-Invocation
<u>Tracepoint</u>	Static tracepoint in DEVHELP.
<u>Minor Code</u>	171 (0X00AB)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 172 (0X00AC)

Description (OS) DevHlp\_SchedClock Post-Invocation

Tracepoint Static tracepoint in DEVHELP.

Minor Code 172 (0X00AC)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 173 (0X00AD)

Description (OS) DevHlp\_SemClear Post-Invocation

Tracepoint Static tracepoint in DEVHELP.

Minor Code 173 (0X00AD)

Trace Groups No groups assigned.

Trace Types No types assigned.

Traced Parameters Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 174 (0X00AE)



<b><u>Description</u></b>	(OS) DevHlp_SemHandle Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	174 (0X00AE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W    SemHandle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 175 (0X00AF)

<b><u>Description</u></b>	(OS) DevHlp_SemRequest Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	175 (0X00AF)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 176 (0X00B0)

<b><u>Description</u></b>	(OS) DevHlp_SendEvent Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	176 (0X00B0)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 177 (0X00B1)

**Description**  
(OS) DevHlp\_SetIRQ Post-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
177 (0X00B1)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 178 (0X00B2)

**Description**  
(OS) DevHlp\_SetROMVector Post-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
178 (0X00B2)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 179 (0X00B3)

<b><u>Description</u></b>	(OS) DevHlp_SetTimer Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	179 (0X00B3)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W

## DEVHELP Major Code: 0X0006 Minor Code: 180 (0X00B4)

<b><u>Description</u></b>	(OS) DevHlp_SortReqPacket Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	180 (0X00B4)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

## DEVHELP Major Code: 0X0006 Minor Code: 181 (0X00B5)

<b><u>Description</u></b>	(OS) DevHlp_TCYield Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	181 (0X00B5)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**  
No parameters traced.

-----

DEVHELP Major Code: 0X0006 Minor Code: 182 (0X00B6)

**Description**  
(OS) DevHlp\_TickCount Post-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
182 (0X00B6)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 183 (0X00B7)

**Description**  
(OS) DevHlp\_Unlock Post-Invocation

**Tracepoint**  
Static tracepoint in DEVHELP.

**Minor Code**  
183 (0X00B7)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 184 (0X00B8)

**Description**

(OS) DevHlp\_UnPhysToVirt Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

184 (0X00B8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 185 (0X00B9)

**Description**

(OS) DevHlp\_UnSetIRQ Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

185 (0X00B9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 186 (0X00BA)

**Description**

(OS) DevHlp\_VerifyAccess Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

186 (0X00BA)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W

-----

DEVHELP Major Code: 0X0006 Minor Code: 188 (0X00BC)

**Description**

(OS) DevHlp\_VirtToPhys Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

188 (0X00BC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W    Physical Address=%D

-----

DEVHELP Major Code: 0X0006 Minor Code: 189 (0X00BD)

**Description**

(OS) DevHlp\_VMAlloc Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

189 (0X00BD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D

-----

DEVHELP Major Code: 0X0006 Minor Code: 190 (0X00BE)

<b><u>Description</u></b>	(OS) DevHlp_VMFree Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	190 (0X00BE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 191 (0X00BF)

<b><u>Description</u></b>	(OS) DevHlp_VMGlobalToProcess Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	191 (0X00BF)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%D    Linear address=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 193 (0X00C1)

<b><u>Description</u></b>	(OS) DevHlp_VMLock Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	193 (0X00C1)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

Return code=%D

-----

DEVHELP Major Code: 0X0006 Minor Code: 194 (0X00C2)

**Description**

(OS) DevHlp\_VMProcessToGlobal Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

194 (0X00C2)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D    Linear address=%D

-----

DEVHELP Major Code: 0X0006 Minor Code: 195 (0X00C3)

**Description**

(OS) DevHlp\_VMUnlock Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

195 (0X00C3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D

-----

DEVHELP Major Code: 0X0006 Minor Code: 197 (0X00C5)



<b><u>Description</u></b>	(OS) DevHlp_Yield Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	197 (0X00C5)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 201 (0X00C9)

<b><u>Description</u></b>	(OS) DevHlp_FreeGDTSelector Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	201 (0X00C9)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 202 (0X00CA)

<b><u>Description</u></b>	(OS) DevHlp_VirtToLin Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	202 (0X00CA)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

Return code=%D    Linear address=%D

-----

**DEVHELP Major Code: 0X0006 Minor Code: 203 (0X00CB)**

**Description**

(OS) DevHlp\_LinToGDTSelector Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

203 (0X00CB)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D

-----

**DEVHELP Major Code: 0X0006 Minor Code: 204 (0X00CC)**

**Description**

(OS) DevHlp\_GetDescInfo Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

204 (0X00CC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D    Attrib. byte=%B    Access byte=%B

Phys. address=%D    Descr. limit=%D

-----

**DEVHELP Major Code: 0X0006 Minor Code: 205 (0X00CD)**

<b><u>Description</u></b>	(OS) DevHlp_LinToPageList Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	205 (0X00CD)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	Return code=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 206 (0X00CE)

<b><u>Description</u></b>	(OS) DevHlp_PageListToLin Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	206 (0X00CE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	Return code=%D    Linear address=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 207 (0X00CF)

<b><u>Description</u></b>	(OS) DevHlp_PageListToGDTSelector Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	207 (0X00CF)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 209 (0X00D1)

**Description**

(OS) DevHlp\_PhysToGDTSel Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

209 (0X00D1)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 211 (0X00D3)

**Description**

(OS) DevHlp\_AllocCtxHook Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

211 (0X00D3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 212 (0X00D4)

<b><u>Description</u></b>	(OS) DevHlp_ArmCtxHook Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	212 (0X00D4)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 213 (0X00D5)

<b><u>Description</u></b>	(OS) DevHlp_FreeCtxHook Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	213 (0X00D5)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 215 (0X00D7)

<b><u>Description</u></b>	(OS) DevHlp_OpenEventSem Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	215 (0X00D7)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D, Sem Handle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 216 (0X00D8)

**Description**

(OS) DevHlp\_CloseEventSem Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

216 (0X00D8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D, Sem Handle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 217 (0X00D9)

**Description**

(OS) DevHlp\_PostEventSem Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

217 (0X00D9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%D, Sem Handle=%D

-----

## DEVHELP Major Code: 0X0006 Minor Code: 218 (0X00DA)

<b>Description</b>	(OS) DevHlp_ResetEventSem Post-Invocation
<b>Tracepoint</b>	Static tracepoint in DEVHELP.
<b>Minor Code</b>	218 (0X00DA)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	Return code=%D, Sem Handle=%D, Post Count=%D

## DEVHELP Major Code: 0X0006 Minor Code: 221 (0X00DD)

<b>Description</b>	(OS) DevHlp_ProcRun2 Post-Invocation
<b>Tracepoint</b>	Static tracepoint in DEVHELP.
<b>Minor Code</b>	221 (0X00DD)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	No parameters traced.

## DEVHELP Major Code: 0X0006 Minor Code: 236 (0X00EC)

<b>Description</b>	(OS) DevHlp_ModifyPriority Post-Invocation
<b>Tracepoint</b>	Static tracepoint in DEVHELP.
<b>Minor Code</b>	236 (0X00EC)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 237 (0X00ED)

**Description**

(OS) DevHlp\_RegisterTmrDD Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

237 (0X00ED)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

pqwTmrRollover=%A pqwTmr=%A

-----

## DEVHELP Major Code: 0X0006 Minor Code: 238 (0X00EE)

**Description**

(OS) DevHlp\_RegisterPerfCtrs Post-Invocation

**Tracepoint**

Static tracepoint in DEVHELP.

**Minor Code**

238 (0X00EE)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 252 (0X00FC)



<b><u>Description</u></b>	(OS) DevHlp_RegisterPDD Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	252 (0X00FC)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## DEVHELP Major Code: 0X0006 Minor Code: 253 (0X00FD)

<b><u>Description</u></b>	(OS) DevHlp_RegisterBeep Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	253 (0X00FD)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Return code=%W

-----

## DEVHELP Major Code: 0X0006 Minor Code: 254 (0X00FE)

<b><u>Description</u></b>	(OS) DevHlp_Beep Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in DEVHELP.
<b><u>Minor Code</u></b>	254 (0X00FE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

Error code=%W

---

## Miscellaneous System Trace Events

The tracepoints for the Miscellaneous major code are identified in the following table. These tracepoints are listed separately from the other kernel tracepoints because they are static tracepoints. They are compiled with the code.

### Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Disk device driver:

Trace events for DISK02 Major Code: 0X0007, sorted by minor code.  
Trace events for DISK02 Major Code: 0X0007, sorted by tracepoint.

---

## Trace Events for DISK02 Major Code: 0X0007, Sorted by Minor Code

00001 (0X0001) (OS) Disk Device Driver Read Pre-Invocation  
00002 (0X0002) (OS) Disk Device Driver Write Pre-Invocation  
00003 (0X0003) (OS) Disk Device Driver WriteVerify Pre-Invocation  
00006 (0X0006) (OS) Disk Device Driver SCB Transfer Pre-Invocation  
32769 (0X8001) (OS) Disk Device Driver Read Post-Invocation  
32770 (0X8002) (OS) Disk Device Driver Write Post-Invocation  
32771 (0X8003) (OS) Disk Device Driver WriteVerify Post-Invocation  
32774 (0X8006) (OS) Disk Device Driver SCB Transfer Post-Invocation

---

## Trace Events for DISK02 Major Code: 0X0007, Sorted by Tracepoint

(OS) Disk Device Driver Read Post-Invocation 32769 (0X8001)  
(OS) Disk Device Driver Read Pre-Invocation 00001 (0X0001)  
(OS) Disk Device Driver SCB Transfer Post-Invocation 32774 (0X8006)  
(OS) Disk Device Driver SCB Transfer Pre-Invocation 00006 (0X0006)  
(OS) Disk Device Driver Write Post-Invocation 32770 (0X8002)  
(OS) Disk Device Driver Write Pre-Invocation 00002 (0X0002)  
(OS) Disk Device Driver WriteVerify Post-Invocation 32771 (0X8003)  
(OS) Disk Device Driver WriteVerify Pre-Invocation 00003 (0X0003)

---

DISK02 Major Code: 0X0007 Minor Code: 1 (0X0001)

<u>Description</u>	(OS) Disk Device Driver Read Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DISK02.
<u>Minor Code</u>	1 (0X0001)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Drive ID=%W Relative Block Addr=%Q Number of Sectors=%W Buffer Addr=%D

DISK02 Major Code: 0X0007 Minor Code: 2 (0X0002)

<u>Description</u>	(OS) Disk Device Driver Write Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DISK02.
<u>Minor Code</u>	2 (0X0002)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Drive ID=%W Relative Block Addr=%Q Number of Sectors=%W Buffer Addr=%D

DISK02 Major Code: 0X0007 Minor Code: 3 (0X0003)

<u>Description</u>	(OS) Disk Device Driver WriteVerify Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in DISK02.
<u>Minor Code</u>	

3 (0X0003)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Drive ID=%W Relative Block Addr=%Q

Number of Sectors=%W Buffer Addr=%D

-----

## DISK02 Major Code: 0X0007 Minor Code: 6 (0X0006)

**Description**

(OS) Disk Device Driver SCB Transfer Pre-Invocation

**Tracepoint**

Static tracepoint in DISK02.

**Minor Code**

6 (0X0006)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Drive ID=%W Relative Block Addr=%Q

Number of Sectors=%W

Req Offset=%W Req Selector=%W Req List Entry Offset=%D

Scatter/Gather Count=%W Command=%W

-----

## DISK02 Major Code: 0X0007 Minor Code: 32769 (0X8001)

**Description**

(OS) Disk Device Driver Read Post-Invocation

**Tracepoint**

Static tracepoint in DISK02.

**Minor Code**

32769 (0X8001)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Drive ID=%W Completion Status=%W

-----

DISK02 Major Code: 0X0007 Minor Code: 32770 (0X8002)

**Description**

(OS) Disk Device Driver Write Post-Invocation

**Tracepoint**

Static tracepoint in DISK02.

**Minor Code**

32770 (0X8002)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Drive ID=%W Completion Status=%W

-----

DISK02 Major Code: 0X0007 Minor Code: 32771 (0X8003)

**Description**

(OS) Disk Device Driver WriteVerify Post-Invocation

**Tracepoint**

Static tracepoint in DISK02.

**Minor Code**

32771 (0X8003)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Drive ID=%W Completion Status=%W

-----

DISK02 Major Code: 0X0007 Minor Code: 32774 (0X8006)

**Description** (OS) Disk Device Driver SCB Transfer Post-Invocation

**Tracepoint** Static tracepoint in DISK02.

**Minor Code** 32774 (0X8006)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
Drive ID=%W Completion Status=%W

DASD manager device driver:

Trace events for OS2DASD Major Code: 0X0007, sorted by minor code.  
Trace events for OS2DASD Major Code: 0X0007, sorted by tracepoint.

---

## Trace Events for OS2DASD Major Code: 0X0007, Sorted by Minor Code

00008 (0X0008) (OS) DASD Manager Strategy-1 Read/Write/Verify Pre-Invocation  
00009 (0X0009) (OS) DASD Manager IOCTL Pre-Invocation  
00010 (0X000A) (OS) DASD Manager Strategy-2 Request List Header Pre-Invocation  
00011 (0X000B) (OS) DASD Manager Strategy-2 Read/Write/Verify Pre-Invocation  
00012 (0X000C) (OS) DASD Manager IORB Pre-Invocation  
00136 (0X0088) (OS) DASD Manager Strategy-1 Read/Write/Verify Post-Invocation  
00137 (0X0089) (OS) DASD Manager IOCTL Post-Invocation  
00138 (0X008A) (OS) DASD Manager Strategy-2 Request List Header Post-Invocation  
00139 (0X008B) (OS) DASD Manager Strategy-2 Read/Write/Verify Post-Invocation  
00140 (0X008C) (OS) DASD Manager IORB Post-Invocation

---

## Trace Events for OS2DASD Major Code: 0X0007, Sorted by Tracepoint

(OS) DASD Manager IOCTL Post-Invocation 00137 (0X0089)  
(OS) DASD Manager IOCTL Pre-Invocation 00009 (0X0009)  
(OS) DASD Manager IORB Post-Invocation 00140 (0X008C)  
(OS) DASD Manager IORB Pre-Invocation 00012 (0X000C)  
(OS) DASD Manager Strategy-1 Read/Write/Verify Post-Invocation 00136 (0X0088)  
(OS) DASD Manager Strategy-1 Read/Write/Verify Pre-Invocation 00008 (0X0008)  
(OS) DASD Manager Strategy-2 Read/Write/Verify Post-Invocation 00139 (0X008B)  
(OS) DASD Manager Strategy-2 Read/Write/Verify Pre-Invocation 00011 (0X000B)  
(OS) DASD Manager Strategy-2 Request List Header Post-Invocation 00138 (0X008A)  
(OS) DASD Manager Strategy-2 Request List Header Pre-Invocation 00010 (0X000A)

-----

## OS2DASD Major Code: 0X0007 Minor Code: 8 (0X0008)

<u>Description</u>	(OS) DASD Manager Strategy-1 Read/Write/Verify Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in OS2DASD.
<u>Minor Code</u>	8 (0X0008)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<p>pRequest=%A Unit=%B %l1 Drive=%S Cmd=%B %l1 %S %l3 Prty=%B %l1</p> <p>cSGList=%W RBA=%F Sectors=%F %l4</p>

-----

## OS2DASD Major Code: 0X0007 Minor Code: 9 (0X0009)

<u>Description</u>	(OS) DASD Manager IOCTL Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in OS2DASD.
<u>Minor Code</u>	9 (0X0009)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<p>pRequest=%A Unit=%B %l1 Drive=%S Cat/Func=%B %B %S %l3 Prty=%B %l1</p> <p>cSGList=%W CH=%W %W Sectors=%F %l4</p>

-----

## OS2DASD Major Code: 0X0007 Minor Code: 10 (0X000A)

<u>Description</u>	(OS) DASD Manager Strategy-2 Request List Header Pre-Invocation
--------------------	---

<u>Tracepoint</u>	Static tracepoint in OS2DASD.
<u>Minor Code</u>	10 (0X000A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	pRLH=%A Count=%W Unit=%B %I1 Drive=%S Ctrl=%W

-----

## OS2DASD Major Code: 0X0007 Minor Code: 11 (0X000B)

<u>Description</u>	(OS) DASD Manager Strategy-2 Read/Write/Verify Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in OS2DASD.
<u>Minor Code</u>	11 (0X000B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	pRequest=%A Unit=%B %I1 Drive=%S Cmd=%B %I1 %S %I1 Ctrl=%W Prty=%B %I1 cSGList=%W RBA=%F Sectors=%F pRLH=%W %I2

-----

## OS2DASD Major Code: 0X0007 Minor Code: 12 (0X000C)

<u>Description</u>	(OS) DASD Manager IORB Pre-Invocation
<u>Tracepoint</u>	Static tracepoint in OS2DASD.
<u>Minor Code</u>	12 (0X000C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.



**Traced Parameters**

pRequest=%A Unit=%B %l1 %l2 Cmd=%B %B %S %l1 Ctrl=%W %l2  
cSGList=%W RBA=%F Sectors=%F %l4

-----

OS2DASD Major Code: 0X0007 Minor Code: 136 (0X0088)

**Description**

(OS) DASD Manager Strategy-1 Read/Write/Verify Post-Invocation

**Tracepoint**

Static tracepoint in OS2DASD.

**Minor Code**

136 (0X0088)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

pRequest=%A Status=%B %l1 ErrorCode=%B %l1 SectorsDone=%F

-----

OS2DASD Major Code: 0X0007 Minor Code: 137 (0X0089)

**Description**

(OS) DASD Manager IOCTL Post-Invocation

**Tracepoint**

Static tracepoint in OS2DASD.

**Minor Code**

137 (0X0089)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

pRequest=%A Status=%B %l1 ErrorCode=%B %l5

-----

OS2DASD Major Code: 0X0007 Minor Code: 138 (0X008A)

<b><u>Description</u></b>	(OS) DASD Manager Strategy-2 Request List Header Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in OS2DASD.
<b><u>Minor Code</u></b>	138 (0X008A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	pRLH=%A DoneCount=%W Status=%B %I1

-----

## OS2DASD Major Code: 0X0007 Minor Code: 139 (0X008B)

<b><u>Description</u></b>	(OS) DASD Manager Strategy-2 Read/Write/Verify Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in OS2DASD.
<b><u>Minor Code</u></b>	139 (0X008B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	pRequest=%A Status=%B %I1 ErrorCode=%B %I1 SectorsDone=%F

-----

## OS2DASD Major Code: 0X0007 Minor Code: 140 (0X008C)

<b><u>Description</u></b>	(OS) DASD Manager IORB Post-Invocation
<b><u>Tracepoint</u></b>	Static tracepoint in OS2DASD.
<b><u>Minor Code</u></b>	140 (0X008C)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

pRequest=%A Status=%W ErrorCode=%W SectorsDone=%F

Resource Manager Device Driver Trace Events

The tracepoints for the Miscellaneous major code are identified in the following table. These tracepoints are listed separately from the other kernel tracepoints because they are static tracepoints. They are compiled with the code.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Resource manager device driver:

Trace events for RESOURCE major code: 0X0008, sorted by minor code.  
Trace events for RESOURCE major code: 0X0008 ,sorted by tracepoint.

Trace Events for RESOURCE Major Code: 0X0008, Sorted by Minor Code

00001 (0X0001) (OS) Resource Manager Allocate  
00002 (0X0002) (OS) Resource Manager Allocate  
00003 (0X0003) (OS) Resource Manager Allocate  
00017 (0X0011) (OS) Resource Manager Deallocate  
00018 (0X0012) (OS) Resource Manager Deallocate  
00019 (0X0013) (OS) Resource Manager Deallocate

Trace Events for RESOURCE Major Code: 0X0008, Sorted by Tracepoint

(OS) Resource Manager Deallocate 00017 (0X0011)  
(OS) Resource Manager Deallocate 00018 (0X0012)  
(OS) Resource Manager Deallocate 00019 (0X0013)  
(OS) Resource Manager Allocate 00001 (0X0001)  
(OS) Resource Manager Allocate 00002 (0X0002)  
(OS) Resource Manager Allocate 00003 (0X0003)

RESOURCE Major Code: 0X0008 Minor Code: 1 (0X0001)

<b><u>Description</u></b>	(OS) Resource Manager Allocate
<b><u>Tracepoint</u></b>	Static trace point in RESOURCE.
<b><u>Minor Code</u></b>	1 (0X0001)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Driver=%s %s=%w%w Flags=%s Rc=%w

## RESOURCE Major Code: 0X0008 Minor Code: 2 (0X0002)

<b><u>Description</u></b>	(OS) Resource Manager Allocate
<b><u>Tracepoint</u></b>	Static trace point in RESOURCE.
<b><u>Minor Code</u></b>	2 (0X0002)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Driver=%s %s=%w Flags=%s Rc=%w

## RESOURCE Major Code: 0X0008 Minor Code: 3 (0X0003)

<b><u>Description</u></b>	(OS) Resource Manager Allocate
<b><u>Tracepoint</u></b>	Static trace point in RESOURCE.
<b><u>Minor Code</u></b>	

3 (0X0003)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Driver=%s

%s=%f%f Flags=%s Rc=%w

---

## RESOURCE Major Code: 0X0008 Minor Code: 17 (0X0011)

**Description**

(OS) Resource Manager Deallocate

**Tracepoint**

Static trace point in RESOURCE.

**Minor Code**

17 (0X0011)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Driver=%s

%s=%w%w Flags=%s Rc=%w

---

## RESOURCE Major Code: 0X0008 Minor Code: 18 (0X0012)

**Description**

(OS) Resource Manager Deallocate

**Tracepoint**

Static trace point in RESOURCE.

**Minor Code**

18 (0X0012)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

Driver=%s  
%s=%w Flags=%s Rc=%w

RESOURCE Major Code: 0X0008 Minor Code: 19 (0X0013)

Description	(OS) Resource Manager Deallocate
Tracepoint	Static trace point in RESOURCE.
Minor Code	19 (0X0013)
Trace Groups	No groups assigned.
Trace Types	No types assigned.
Traced Parameters	Driver=%s %s=%f%f Flags=%s Rc=%w

DOSCALL1.DLL Trace Events

The tracepoints for the DOSCALL1.DLL kernel services major code are identified in the following tables. These tracepoints are dynamic tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for DOSCALL1 Major Code: 0X0010, sorted by minor code.](#)  
[Trace events for DOSCALL1 Major Code: 0X0010 ,sorted by tracepoint.](#)  
[Kernel API Tracepoints Indirected Via DOSCALL1.](#)  
[QUECALLS API Tracepoints Indirected Via DOSCALL1.](#)

Trace Events for DOSCALL1 Major Code: 0X0010, Sorted by Minor Code

[00008 \(0X0008\) \(OS\) DosFSRamSemClear Pre-Invocation](#)  
[00009 \(0X0009\) \(OS\) DosFSRamSemRequest Pre-Invocation](#)  
[00014 \(0X000E\) \(OS\) DosReadAsync Pre-Invocation](#)  
[00015 \(0X000F\) \(OS\) DosScanEnv Pre-Invocation](#)

```

00016 (0X0010) (OS) DosSearchPath Pre-Invocation
00017 (0X0011) (OS) DosSemClear Pre-Invocation
00018 (0X0012) (OS) DosSemRequest Pre-Invocation
00019 (0X0013) (OS) DosSemSet Pre-Invocation
00020 (0X0014) (OS) DosSemWait Pre-Invocation
00021 (0X0015) (OS) DosSetCp Pre-Invocation
00022 (0X0016) (OS) DosSetProcCp Pre-Invocation
00023 (0X0017) (OS) DosSubAlloc Pre-Invocation
00024 (0X0018) (OS) Dos32SubAlloc Pre-Invocation
00025 (0X0019) (OS) DosSubFree Pre-Invocation
00026 (0X001A) (OS) Dos32SubFree Pre-Invocation
00027 (0X001B) (OS) DosSubSet Pre-Invocation
00028 (0X001C) (OS) Dos32SubSet Pre-Invocation
00030 (0X001E) (OS) DosWriteAsync Pre-Invocation
00032 (0X0020) (OS) DosErrClass Pre-Invocation
00033 (0X0021) (OS) DosQAppType Pre-Invocation
00034 (0X0022) (OS) Dos32SetExceptionHandler Pre-Invocation
00035 (0X0023) (OS) Dos32RaiseException Pre-Invocation
00036 (0X0024) (OS) Dos32UnsetExceptionHandler Pre-Invocation
00037 (0X0025) (OS) Dos32UnwindException Pre-Invocation
00257 (0X0101) (OS) DosGetMessage Pre-Invocation
00258 (0X0102) (OS) DosInsMessage Pre-Invocation
00259 (0X0103) (OS) DosPutMessage Pre-Invocation
00361 (0X0169) (OS) Dos32R3ExceptionHandler Pre-Invocation
00362 (0X016A) (OS) Dos32ExceptionCallback Pre-Invocation
00363 (0X016B) (OS) xcptExecuteUserExceptionHandler Pre-Invocation
32776 (0X8008) (OS) DosFSRamSemClear Post-Invocation
32777 (0X8009) (OS) DosFSRamSemRequest Post-Invocation
32782 (0X800E) (OS) DosReadAsync Post-Invocation
32783 (0X800F) (OS) DosScanEnv Post-Invocation
32784 (0X8010) (OS) DosSearchPath Post-Invocation
32785 (0X8011) (OS) DosSemClear Post-Invocation
32786 (0X8012) (OS) DosSemRequest Post-Invocation
32787 (0X8013) (OS) DosSemSet Post-Invocation
32788 (0X8014) (OS) DosSemWait Post-Invocation
32789 (0X8015) (OS) DosSetCp Post-Invocation
32790 (0X8016) (OS) DosSetProcCp Post-Invocation
32791 (0X8017) (OS) DosSubAlloc Post-Invocation
32792 (0X8018) (OS) Dos32SubAlloc Post-Invocation
32793 (0X8019) (OS) DosSubFree Post-Invocation
32794 (0X801A) (OS) Dos32SubFree Post-Invocation
32795 (0X801B) (OS) DosSubSet Post-Invocation
32796 (0X801C) (OS) Dos32SubSet Post-Invocation
32798 (0X801E) (OS) DosWriteAsync Post-Invocation
32800 (0X8020) (OS) DosErrClass Post-Invocation
32801 (0X8021) (OS) DosQAppType Post-Invocation
32802 (0X8022) (OS) Dos32SetExceptionHandler Post-Invocation
32804 (0X8024) (OS) Dos32UnsetExceptionHandler Post-Invocation
33025 (0X8101) (OS) DosGetMessage Post-Invocation
33026 (0X8102) (OS) DosInsMessage Post-Invocation
33027 (0X8103) (OS) DosPutMessage Post-Invocation
33028 (0X8104) (OS) UniThunk 32:16 Return
33029 (0X8105) (OS) UniThunk 16:32 Return

```

-----

## Trace Events for DOSCALL1 Major Code: 0X0010, Sorted by Tracepoint

```

DOS32EXCEPTIONCALLBACK 00362 (0X016A)
DOS32R3EXCEPTIONDISPATCHER 00361 (0X0169)
DOS32RAISEEXCEPTION 00035 (0X0023)
DOS32SETEXCEPTIONHANDLER 00034 (0X0022)
Dos32SubAlloc Post-Invocation 32792 (0X8018)
Dos32SubAlloc Pre-Invocation 00024 (0X0018)
Dos32SubFree Post-Invocation 32794 (0X801A)

```

Dos32SubFree Pre-Invocation 00026 (0X001A)  
 Dos32SubSet Post-Invocation 32796 (0X801C)  
 Dos32SubSet Pre-Invocation 00028 (0X001C)  
 DOS32UNSETEXCEPTIONHANDLER 00036 (0X0024)  
 DOS32UNWINDEXCEPTION 00037 (0X0025)  
 DOSFSRAMSEMCLEAR 00008 (0X0008)  
 DOSFSRAMSEMREQUEST 00009 (0X0009)  
 DOSREADASYNC 00014 (0X000E)  
 DOSSCANENV 00015 (0X000F)  
 DOSSEARCHPATH 00016 (0X0010)  
 DOSSEMCLEAR 00017 (0X0011)  
 DOSSEMREQUEST 00018 (0X0012)  
 DOSSEMSET 00019 (0X0013)  
 DOSSEMWAIT 00020 (0X0014)  
 DOSSETCP 00021 (0X0015)  
 DOSSETPROCCP 00022 (0X0016)  
 DOSSUBSET 00027 (0X001B)  
 DOSWRITEASYNC 00030 (0X001E)  
 POSTGETMSG 33025 (0X8101)  
 POSTINMSG 33026 (0X8102)  
 POSTPUTMSG 33027 (0X8103)  
 PREGETMSG 00257 (0X0101)  
 PREINMSG 00258 (0X0102)  
 PREPUTMSG 00259 (0X0103)  
 postDOS32SETEXCEPTIONHANDLER 32802 (0X8022)  
 postDOS32UNSETEXCEPTIONHANDLER 32804 (0X8024)  
 postDOSERRCLASS 32800 (0X8020)  
 postDOSFSRAMSEMCLEAR 32776 (0X8008)  
 postDOSFSRAMSEMREQUEST 32777 (0X8009)  
 postDOSQAPPTYPE 32801 (0X8021)  
 postDOSREADASYNC 32782 (0X800E)  
 postDOSSCANENV 32783 (0X800F)  
 postDOSSEARCHPATH 32784 (0X8010)  
 postDOSSEMCLEAR 32785 (0X8011)  
 postDOSSEMREQUEST 32786 (0X8012)  
 postDOSSEMSET 32787 (0X8013)  
 postDOSSEMWAIT 32788 (0X8014)  
 postDOSSETCP 32789 (0X8015)  
 postDOSSETPROCCP 32790 (0X8016)  
 postDOSSUBALLOC 32791 (0X8017)  
 postDOSSUBFREE 32793 (0X8019)  
 postDOSSUBSET 32795 (0X801B)  
 postDOSWRITEASYNC 32798 (0X801E)  
 preDOSERRCLASS 00032 (0X0020)  
 preDOSQAPPTYPE 00033 (0X0021)  
 preDOSSUBALLOC 00023 (0X0017)  
 preDOSSUBFREE 00025 (0X0019)  
 xcptExecuteUserExceptionHandler 00363 (0X016B)  
 UniThunk 16:32 Return33029 (0X8105)  
 UniThunk 32:16 Return33028 (0X8104)

## DOSCALL1 Major Code: 0X0010 Minor Code: 8 (0X0008)

### Description

(OS) DosFSRamSemClear Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: DOSCALL1.DOSFSRAMSEMCLEAR

### Minor Code

8 (0X0008)

### Trace Groups

SEM



**Trace Types**  
PRE

**Traced Parameters**  
  
Sem Ptr = %P %A

**Note:** Parameters are trace with OS/2 Warp V3.0 fix pack 35 or OS/2 Warp V4.0 fix pack 10 or later.

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 9 (0X0009)

**Description**  
(OS) DosFSRamSemRequest Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: DOSCALL1.DOSFSRAMSEMREQUEST

**Minor Code**  
9 (0X0009)

**Trace Groups**  
SEM

**Trace Types**  
PRE

**Traced Parameters**  
  
Timeout = %P %F Sem Ptr = %P %A

**Note:** Parameters are trace with OS/2 Warp V3.0 fix pack 35 or OS/2 Warp V4.0 fix pack 10 or later.

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 14 (0X000E)

**Description**  
(OS) DosReadAsync Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: DOSCALL1.DOSREADASYNC

**Minor Code**  
14 (0X000E)

**Trace Groups**  
FS

**Trace Types**  
PRE

**Traced Parameters**  
  
Sem at = %A Handle = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 15 (0X000F)

<u>Description</u>	(OS) DosScanEnv Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSSCANENV
<u>Minor Code</u>	15 (0X000F)
<u>Trace Groups</u>	LNK
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Current Env = %S
	Env Pointer = %A

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 16 (0X0010)

<u>Description</u>	(OS) DosSearchPath Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEARCHPATH
<u>Minor Code</u>	16 (0X0010)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Control = %W
	Path = %S
	File = %S

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 17 (0X0011)

<u>Description</u>	(OS) DosSemClear Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMCLEAR
<u>Minor Code</u>	17 (0X0011)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Sem Handle = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 18 (0X0012)

<u>Description</u>	(OS) DosSemRequest Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMREQUEST
<u>Minor Code</u>	18 (0X0012)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Sem Handle = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 19 (0X0013)

<u>Description</u>	(OS) DosSemSet Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMSET
<u>Minor Code</u>	19 (0X0013)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	PRE

**Traced Parameters**

Sem Handle = %D

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 20 (0X0014)

**Description**

(OS) DosSemWait Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.DOSSEMWAIT

**Minor Code**

20 (0X0014)

**Trace Groups**

SEM

**Trace Types**

PRE

**Traced Parameters**

Sem Handle = %D

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 21 (0X0015)

**Description**

(OS) DosSetCp Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.DOSSETCP

**Minor Code**

21 (0X0015)

**Trace Groups**

NLS

**Trace Types**

PRE

**Traced Parameters**

Code Page Id= %W

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 22 (0X0016)

<b><u>Description</u></b>	(OS) DosSetProcCp Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSSETPROCCP
<b><u>Minor Code</u></b>	22 (0X0016)
<b><u>Trace Groups</u></b>	NLS
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Code Page Id= %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 23 (0X0017)

<b><u>Description</u></b>	(OS) DosSubAlloc Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: DOSCALL1.preDOSSUBALLOC
<b><u>Minor Code</u></b>	23 (0X0017)
<b><u>Trace Groups</u></b>	MSP
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Selector = %W Size = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 24 (0X0018)

<b><u>Description</u></b>	(OS) Dos32SubAlloc Pre-Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.
<b><u>Minor Code</u></b>	24 (0X0018)
<b><u>Trace Groups</u></b>	MSP
<b><u>Trace Types</u></b>	PRE

**Traced Parameters**

pHdr = %F Size = %D

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 25 (0X0019)

**Description**

(OS) DosSubFree Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.preDOSSUBFREE

**Minor Code**

25 (0X0019)

**Trace Groups**

MSP

**Trace Types**

PRE

**Traced Parameters**

Selector = %W Offset = %W

Size = %W

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 26 (0X001A)

**Description**

(OS) Dos32SubFree Pre-Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

**Minor Code**

26 (0X001A)

**Trace Groups**

MSP

**Trace Types**

PRE

**Traced Parameters**

pHdr = %F Offset = %F

Size = %D

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 27 (0X001B)

<b><u>Description</u></b>	(OS) DosSubSet Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSSUBSET
<b><u>Minor Code</u></b>	27 (0X001B)
<b><u>Trace Groups</u></b>	MSP
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>Selector = %W Flags = %W</p> <p>Size = %W</p>

## DOSCALL1 Major Code: 0X0010 Minor Code: 28 (0X001C)

<b><u>Description</u></b>	(OS) Dos32SubSet Pre-Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.
<b><u>Minor Code</u></b>	28 (0X001C)
<b><u>Trace Groups</u></b>	MSP
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>pHdr = %F Flags = %D</p> <p>Size = %D</p>

## DOSCALL1 Major Code: 0X0010 Minor Code: 30 (0X001E)

<b><u>Description</u></b>	(OS) DosWriteAsync Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: DOSCALL1.DOSWRITEASYNC
<b><u>Minor Code</u></b>	

30 (0X001E)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Sem at = %A Handle = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32 (0X0020)

**Description**

(OS) DosErrClass Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.preDOSERRCLASS

**Minor Code**

32 (0X0020)

**Trace Groups**

FS

**Trace Types**

PRE

**Traced Parameters**

Error Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 33 (0X0021)

**Description**

(OS) DosQAppType Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.preDOSQAPPTYPE

**Minor Code**

33 (0X0021)

**Trace Groups**

LDR

**Trace Types**

PRE

**Traced Parameters**

File = %S



-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 34 (0X0022)

<u>Description</u>	(OS) Dos32SetExceptionHandler Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOS32SETEXCEPTIONHANDLER
<u>Minor Code</u>	34 (0X0022)
<u>Trace Groups</u>	EXMG
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ExcHandlerStructAddress = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 35 (0X0023)

<u>Description</u>	(OS) Dos32RaiseException Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOS32RAISEEXCEPTION
<u>Minor Code</u>	35 (0X0023)
<u>Trace Groups</u>	EXMG
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Exception Report Record: Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F Exception Address=%F Parameter Count=%F P1=%F P2=%F P3=%F P4=%F

**Note:** Parameters are traced only with OS/2 Warp V3.0 fix pack 35 or OS/2 Warp V4.0 fix pack 10 or later.

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 36 (0X0024)

<u>Description</u>	(OS) Dos32UnsetExceptionHandler Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOS32UNSETEXCEPTIONHANDLER
<u>Minor Code</u>	36 (0X0024)
<u>Trace Groups</u>	EXMG
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ExcHandlerStructAddress = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 37 (0X0025)

<u>Description</u>	(OS) Dos32UnwindException Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.DOS32UNWINDEXCEPTION
<u>Minor Code</u>	37 (0X0025)
<u>Trace Groups</u>	EXMG
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ExcHandlerStructAddress = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 257 (0X0101)

<u>Description</u>	(OS) DosGetMessage Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.PREGETMSG
<u>Minor Code</u>	257 (0X0101)
<u>Trace Groups</u>	MSG

Trace Types  
PRE

Traced Parameters  
  
MsgNumber = %W  
Filename = %S

DOSCALL1 Major Code: 0X0010 Minor Code: 258 (0X0102)

Description  
(OS) DosInsMessage Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: DOSCALL1.PREINSMMSG

Minor Code  
258 (0X0102)

Trace Groups  
MSG

Trace Types  
PRE

Traced Parameters  
  
IV Count = %W  
Length of Input Message %W

DOSCALL1 Major Code: 0X0010 Minor Code: 259 (0X0103)

Description  
(OS) DosPutMessage Pre-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: DOSCALL1.PREPUTMSG

Minor Code  
259 (0X0103)

Trace Groups  
MSG

Trace Types  
PRE

Traced Parameters  
  
File Handle = %W  
Message Length = %W

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 32776  
(0X8008)

<u>Description</u>	(OS) DosFSRamSemClear Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSFSRAMSEMCLEAR
<u>Minor Code</u>	32776 (0X8008)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %W

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 32777  
(0X8009)

<u>Description</u>	(OS) DosFSRamSemRequest Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSFSRAMSEMREQUEST
<u>Minor Code</u>	32777 (0X8009)
<u>Trace Groups</u>	SEM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %W

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 32782  
(0X800E)

<u>Description</u>	(OS) DosReadAsync Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSREADASYNC
<u>Minor Code</u>	32782 (0X800E)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Bytes Read = %W Return Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32783 (0X800F)

<u>Description</u>	(OS) DosScanEnv Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSCANENV
<u>Minor Code</u>	32783 (0X800F)
<u>Trace Groups</u>	LNK
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32784 (0X8010)

<u>Description</u>	(OS) DosSearchPath Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEARCHPATH
<u>Minor Code</u>	

32784 (0X8010)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32785 (0X8011)

**Description**

(OS) DosSemClear Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMCLEAR

**Minor Code**

32785 (0X8011)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32786 (0X8012)

**Description**

(OS) DosSemRequest Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMREQUEST

**Minor Code**

32786 (0X8012)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

---

## DOSCALL1 Major Code: 0X0010 Minor Code: 32787 (0X8013)

**Description**

(OS) DosSemSet Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMSET

**Minor Code**

32787 (0X8013)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

---

## DOSCALL1 Major Code: 0X0010 Minor Code: 32788 (0X8014)

**Description**

(OS) DosSemWait Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSEMWAIT

**Minor Code**

32788 (0X8014)

**Trace Groups**

SEM

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

---

## DOSCALL1 Major Code: 0X0010 Minor Code: 32789

# (0X8015)

<u>Description</u>	(OS) DosSetCp Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSETCP
<u>Minor Code</u>	32789 (0X8015)
<u>Trace Groups</u>	NLS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code= %W

## DOSCALL1 Major Code: 0X0010 Minor Code: 32790 (0X8016)

<u>Description</u>	(OS) DosSetProcCp Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSETPROCCP
<u>Minor Code</u>	32790 (0X8016)
<u>Trace Groups</u>	NLS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code= %W

## DOSCALL1 Major Code: 0X0010 Minor Code: 32791 (0X8017)

<u>Description</u>	(OS) DosSubAlloc Post-Invocation
<u>Tracepoint</u>	



Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSUBALLOC

**Minor Code**

32791 (0X8017)

**Trace Groups**

MSP

**Trace Types**

POST

**Traced Parameters**

Offset = %W Return Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32792 (0X8018)

**Description**

(OS) Dos32SubAlloc Post-Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

**Minor Code**

32792 (0X8018)

**Trace Groups**

MSP

**Trace Types**

POST

**Traced Parameters**

Offset = %F Return Code = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32793 (0X8019)

**Description**

(OS) DosSubFree Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSUBFREE

**Minor Code**

32793 (0X8019)

**Trace Groups**

MSP

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 32794  
(0X801A)

**Description**

(OS) Dos32SubFree Post-Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.

**Minor Code**

32794 (0X801A)

**Trace Groups**

MSP

**Trace Types**

POST

**Traced Parameters**

Return Code = %D

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 32795  
(0X801B)

**Description**

(OS) DosSubSet Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSSUBSET

**Minor Code**

32795 (0X801B)

**Trace Groups**

MSP

**Trace Types**

POST

**Traced Parameters**

Return Code = %W

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 32796

## (0X801C)

<u>Description</u>	(OS) Dos32SubSet Post-Invocation
<u>Tracepoint</u>	Source line defined dynamic tracepoint: @msp32.c in DOSCALL1.
<u>Minor Code</u>	32796 (0X801C)
<u>Trace Groups</u>	MSP
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32798 (0X801E)

<u>Description</u>	(OS) DosWriteAsync Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: DOSCALL1.postDOSWRITEASYNC
<u>Minor Code</u>	32798 (0X801E)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Bytes Written = %W Return Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32800 (0X8020)

<u>Description</u>	(OS) DosErrClass Post-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSERRCLASS

**Minor Code**

32800 (0X8020)

**Trace Groups**

FS

**Trace Types**

POST

**Traced Parameters**

Class = %W Action = %W

Locus = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32801 (0X8021)

**Description**

(OS) DosQAppType Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOSQAPPTYPE

**Minor Code**

32801 (0X8021)

**Trace Groups**

LDR

**Trace Types**

POST

**Traced Parameters**

Type = %W Return Code = %W

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32802 (0X8022)

**Description**

(OS) Dos32SetExceptionHandler Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.postDOS32SETEXCEPTIONHANDLER

**Minor Code**

32802 (0X8022)

**Trace Groups**

EXMG

**Trace Types** POST

**Traced Parameters** No parameters traced.

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 32804 (0X8024)

**Description** (OS) Dos32UnsetExceptionHandler Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: DOSCALL1.postDOS32UNSETEXCEPTIONHANDLER

**Minor Code** 32804 (0X8024)

**Trace Groups** EXMG

**Trace Types** POST

**Traced Parameters** Return Code = %D

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 33025 (0X8101)

**Description** (OS) DosGetMessage Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: DOSCALL1.POSTGETMSG

**Minor Code** 33025 (0X8101)

**Trace Groups** MSG

**Trace Types** POST

**Traced Parameters** Return Code = %W  
Message Length = %W

# DOSCALL1 Major Code: 0X0010 Minor Code: 33026 (0X8102)

Description	(OS) DosInsMessage Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: DOSCALL1.POSTINSMSG
Minor Code	33026 (0X8102)
Trace Groups	MSG
Trace Types	POST
Traced Parameters	
	Return Code = %W
	Message Length = %W

-----

# DOSCALL1 Major Code: 0X0010 Minor Code: 33027 (0X8103)

Description	(OS) DosPutMessage Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: DOSCALL1.POSTPUTMSG
Minor Code	33027 (0X8103)
Trace Groups	MSG
Trace Types	POST
Traced Parameters	
	Return Code = %W

-----

# DOSCALL1 Major Code: 0X0010 Minor Code: 361 (0X0169)

**Description**

(OS) Dos32R3ExceptionDispatcher Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.DOS32R3EXCEPTIONDISPATCHER

**Minor Code**

361 (0X0169)

**Trace Groups**

EXMG

**Trace Types**

PRE, API

**Traced Parameters**

Return address=%P %F Trap Number=%F pRepRec=%F pContext=%F

Exception Report Record:

Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F

Exception Address=%F Parameter Count=%F

P1=%F P2=%F P3=%F P4=%F

Context Record:

ContextFlags=%P %F

FPU\_CNTRL=%W %I2 FPU\_STATUS=%W %I2 FPU\_TAG=%W %I2

FPU\_IP=%F FPU\_CS=%W FPU\_OPCODE=%W FPU\_DATAOFFS=%F %I4

FP Register Stack in Lo-DWORD Hi-DWORD Sign/Exp form:

R0=%F %F %W R1=%F %F %W

R2=%F %F %W R3=%F %F %W

R4=%F %F %W R5=%F %F %W

R6=%F %F %W R7=%F %F %W

GS=%W %I2 FS=%W %I2 ES=%W %I2 DS=%W %I2

EDI=%F ESI=%F EAX=%F EBX=%F ECX=%F EDX=%F

EBP=%F EIP=%F CS=%W %I2 EFLAGS=%F ESP=%F SS=%W %I2

**Note:**

This tracepoint is available with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 or later.

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 362 (0X016A)

**Description**

(OS) Dos32ExceptionCallback Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.DOS32EXCEPTIONCALLBACK

**Minor Code**

362 (0X016A)

**Trace Groups**

EXMG

**Trace Types**

PRE, API

**Traced Parameters**

Return address=%P %F Trap Number=%F pRepRec=%F pContext=%F Disposition=%F

Exception Report Record:

Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F

Exception Address=%F Parameter Count=%F

P1=%F P2=%F P3=%F P4=%F

Context Record:

ContextFlags=%P %F

FPU\_CNTRL=%W %I2 FPU\_STATUS=%W %I2 FPU\_TAG=%W %I2

FPU\_IP=%F FPU\_CS=%W FPU\_OPCODE=%W FPU\_DATAOFFS=%F %I4

FP Register Stack in Lo-DWORD Hi-DWORD Sign/Exp form:

R0=%F %F %W R1=%F %F %W

R2=%F %F %W R3=%F %F %W

R4=%F %F %W R5=%F %F %W

R6=%F %F %W R7=%F %F %W

GS=%W %I2 FS=%W %I2 ES=%W %I2 DS=%W %I2

EDI=%F ESI=%F EAX=%F EBX=%F ECX=%F EDX=%F

EBP=%F EIP=%F CS=%W %I2 EFLAGS=%F ESP=%F SS=%W %I2

**Note:**

This tracepoint is available with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 or later.

---

## DOSCALL1 Major Code: 0X0010 Minor Code: 363 (0X016B)

**Description**

(OS) xcptExecuteUserExceptionHandler Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.xcptExecuteUserExceptionHandler

**Minor Code**

363 (0X016B)

**Trace Groups**

EXMG



**Trace Types**

PRE, INT

**Traced Parameters**

Return address=%P %F pRepRec=%F pRegRec=%F pContext=%F pDisContext=%F pNestCatcher=%F

Exception Report Record:

Exception Number=%P %F Handler Flags=%F Nested Exception Report Record=%F

Exception Address=%F Parameter Count=%F

P1=%F P2=%F P3=%F P4=%F

Registration Record:"

pNextRegRec=%P %F pHandler=%F

Context Record:

ContextFlags=%P %F

FPU\_CNTRL=%W %I2 FPU\_STATUS=%W %I2 FPU\_TAG=%W %I2

FPU\_IP=%F FPU\_CS=%W FPU\_OPCODE=%W FPU\_DATAOFFS=%F %I4

FP Register Stack in Lo-DWORD Hi-DWORD Sign/Exp form:

R0=%F %F %W R1=%F %F %W

R2=%F %F %W R3=%F %F %W

R4=%F %F %W R5=%F %F %W

R6=%F %F %W R7=%F %F %W

GS=%W %I2 FS=%W %I2 ES=%W %I2 DS=%W %I2

EDI=%F ESI=%F EAX=%F EBX=%F ECX=%F EDX=%F

EBP=%F EIP=%F CS=%W %I2 EFLAGS=%F ESP=%F SS=%W %I2

**Note:**

This tracepoint is available with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10 or later.

-----

DOSCALL1 Major Code: 0X0010 Minor Code: 33028  
(0X8104)

**Description**

(OS) UniThunk 32:16 Return

**Tracepoint**

Public symbol defined dynamic tracepoint: DOSCALL1.UT16\_RETURN

**Minor Code**

33028 (0X8104)

**Trace Groups**

UT

**Trace Types**  
POST, INT

**Traced Parameters**  
  
Return address = %P %A

**Note:**  
  
This tracepoint is available with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10 or later.

-----

## DOSCALL1 Major Code: 0X0010 Minor Code: 33029 (0X8105)

**Description**  
(OS) UniThunk 16:32 Return

**Tracepoint**  
Public symbol defined dynamic tracepoint: DOSCALL1.UT32\_RETURN

**Minor Code**  
33029 (0X8105)

**Trace Groups**  
UT

**Trace Types**  
POST, INT

**Traced Parameters**  
  
Return address = %P %F

**Note:**  
  
This tracepoint is available with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10 or later.

-----

## Kernel API Tracepoints Indirected Via DOSCALL1

The following table lists pre-invocation tracepoints for Kernel APIs that are indirected via DOSCALL1. These should be trace in conjunction with their corresponding Kernel pre-invocation tracepoint.

<i>DOSCALL1 API</i>	<i>Minor code</i>	<i>Group</i>	<i>Types</i>
DOS32WAITCHILD	262	TK	PRE,API
DOS32BEEP	263	IO	PRE,API
DOS32PHYSICALDISK	264	FS	PRE,API
DOS32SETCP	265	TK	PRE,API
DOS32SETPROCESSCP	266	TK	PRE,API
DOS32SLEEP	267	TK	PRE,API

DOS32DEVCONFIG	268	FS	PRE,API
DOS32GETDATETIME	269	FS	PRE,API
DOS32SETDATETIME	270	TIM	PRE,API
DOS32EXECPGM	271	TIM	PRE,API
DOS32ENTERCRITSEC	272	TK	PRE,API
DOS32EXITCRITSEC	273	TK	PRE,API
DOS32EXIT	274	TK	PRE,API
DOS32KILLPROCESS	275	TK	PRE,API
DOS32SETPRIORITY	276	TK	PRE,API
DOS32RESUMETHREAD	277	TK	PRE,API
DOS32SUSPENDTHREAD	278	TK	PRE,API
DOS32CREATEPIPE	279	PIP	PRE,API
DOS32CALLNPIPE	287	PIP	PRE,API
DOS32CONNECTNPIPE	288	PIP	PRE,API
DOS32DISCONNECTNPIPE	289	PIP	PRE,API
DOS32CREATENPIPE	290	PIP	PRE,API
DOS32PEEKNPIPE	291	PIP	PRE,API
DOS32QUERYNPSTATE	292	PIP	PRE,API
DOS32RAWREADNPIPE	293	PIP	PRE,API
DOS32RAWWRITENPIPE	294	PIP	PRE,API
DOS32QUERYNPPIPEINFO	295	PIP	PRE,API
DOS32QUERYNPPIPESEMSTATE	296	PIP	PRE,API
DOS32SETNPSTATE	297	PIP	PRE,API
DOS32SETNPPIPESEM	298	PIP	PRE,API
DOS32TRANSACTNPIPE	299	PIP	PRE,API
DOS32WAITNPIPE	300	PIP	PRE,API
DOS32RESETBUFFER	301	FS	PRE,API
DOS32SETCURRENTDIR	302	FS	PRE,API
DOS32SETFILEPTR	303	FS	PRE,API
DOS32PROTECTSETFILEPTR	304	FS	PRE,API
DOS32CLOSE	305	FS	PRE,API
DOS32PROTECTCLOSE	306	FS	PRE,API
DOS32COPY	307	FS	PRE,API
DOS32DELETE	308	FS	PRE,API
DOS32FORCEDELETE	309	FS	PRE,API
DOS32DEVIOCTL	310	FS	PRE,API
DOS32DUPHANDLE	311	FS	PRE,API
DOS32EDITNAME	312	FS	PRE,API
DOS32FINDCLOSE	313	FS	PRE,API

DOS32FSATTACH	314	FS	PRE,API
DOS32FSCTL	315	FS	PRE,API
DOS32MOVE	316	FS	PRE,API
DOS32SETFILESIZE	317	FS	PRE,API
DOS32PROTECTSETFILESIZE	318	FS	PRE,API
DOS32QUERYCURRENTDIR	319	FS	PRE,API
DOS32QUERYCURRENTDISK	320	FS	PRE,API
DOS32QUERYFHHSTATE	321	FS	PRE,API
DOS32PROTECTQUERYFHHSTATE	322	FS	PRE,API
DOS32PQUERYFSATTACH	323	FS	PRE,API
DOS32QUERYFSINFO	324	FS	PRE,API
DOS32QUERYHSTYPE	325	FS	PRE,API
DOS32QUERYVERIFY	326	FS	PRE,API
DOS32DELETEDIR	327	FS	PRE,API
DOS32SEARCHPATH	328	FS	PRE,API
DOS32SETDEFAULTDISK	329	FS	PRE,API
DOS32SETFHHSTATE	330	FS	PRE,API
DOS32PROTECTSETFHHSTATE	331	FS	PRE,API
DOS32SETFSINFO	332	FS	PRE,API
DOS32SETMAXFH	333	FS	PRE,API
DOS32SETRELMAXFH	334	FS	PRE,API
DOS32SETVERIFY	335	FS	PRE,API
DOS32ERRCLASS	336	FS	PRE,API
DOS32ERROR	337	TK	PRE,API
DOS32LOADMODULE	338	LDR	PRE,API
DOS32FREEMODULE	339	LDR	PRE,API
DOS32QUERYMODULEHANDLE	340	LDR	PRE,API
DOS32QUERYMODULENAME	341	LDR	PRE,API
DOS32QUERYAPPTYPE	342	LDR	PRE,API
DOS32PFINDNEXT	343	FS	PRE,API
DOS32SHUTDOWN	344	FS	PRE,API
DOS32OPENCHANGENOTIFY	345	FS	PRE,API
DOS32RESETCHANGENOTIFY	346	FS	PRE,API
DOS32CLOSECHANGENOTIFY	347	FS	PRE,API
DOS32CREATESPINLOCK	348	LOCK	PRE,API
DOS32ACQUIRESPINLOCK	349	LOCK	PRE,API
DOS32RELEASESPINLOCK	350	LOCK	PRE,API
DOS32FREESPINLOCK	351	LOCK	PRE,API

-----

# QUECALLS API Tracepoints Indirected Via DOSCALL1

The following table lists pre-invocation tracepoints for QUECALLS.DLL APIs that are indirected via DOSCALL1. These should be trace in conjunction with their corresponding QUECALLS pre-invocation tracepoint.

<i>DOSCALL1 API</i>	<i>Minor code</i>	<i>Group</i>	<i>Types</i>
DOS32CREATEQUEUE	280	QUE	PRE,API
DOS32OPENQUEUE	281	QUE	PRE,API
DOS32CLOSEQUEUE	282	QUE	PRE,API
DOS32PEEKQUEUE	283	QUE	PRE,API
DOS32PURGEQUEUE	284	QUE	PRE,API
DOS32QUERYQUEUE	285	QUE	PRE,API
DOS32WRITEQUEUE	286	QUE	PRE,API
DOS32READQUEUE	352	QUE	PRE,API
DOS16CREATEQUEUE	353	QUE	PRE,API
DOS16OPENQUEUE	354	QUE	PRE,API
DOS16CLOSEQUEUE	355	QUE	PRE,API
DOS16PEEKQUEUE	356	QUE	PRE,API
DOS16PURGEQUEUE	357	QUE	PRE,API
DOS16QUERYQUEUE	358	QUE	PRE,API
DOS16WRITEQUEUE	359	QUE	PRE,API
DOS16READQUEUE	360	QUE	PRE,API

-----

# MONCALLS.DLL Trace Events

The tracepoints for the MONCALLS.DLL services major code are identified in the following tables. These tracepoints are dynamic tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for MONCALLS Major Code: 0X0010, sorted by minor code.](#)  
[Trace events for MONCALLS Major Code: 0X0010 ,sorted by tracepoint.](#)

-----

# Trace Events for MONCALLS Major Code: 0X0010, Sorted by

# Minor Code

00513 (0X0201) (OS) DosMonClose Pre\_Invocation  
00514 (0X0202) (OS) DosMonOpen Pre\_Invocation  
00515 (0X0203) (OS) DosMonRead Pre\_Invocation  
00516 (0X0204) (OS) DosMonReg Pre\_Invocation  
00517 (0X0205) (OS) DosMonWrite Pre\_Invocation  
33281 (0X8201) (OS) DosMonClose Post\_Invocation  
33282 (0X8202) (OS) DosMonOpen Post\_Invocation  
33283 (0X8203) (OS) DosMonRead Post\_Invocation  
33284 (0X8204) (OS) DosMonReg Post\_Invocation  
33285 (0X8205) (OS) DosMonWrite Post\_Invocation

## Trace Events for MONCALLS Major Code: 0X0010, Sorted by Tracepoint

(OS) DosMonClose Post\_Invocation 33281 (0X8201)  
(OS) DosMonClose Pre\_Invocation 00513 (0X0201)  
(OS) DosMonOpen Post\_Invocation 33282 (0X8202)  
(OS) DosMonOpen Pre\_Invocation 00514 (0X0202)  
(OS) DosMonRead Post\_Invocation 33283 (0X8203)  
(OS) DosMonRead Pre\_Invocation 00515 (0X0203)  
(OS) DosMonReg Post\_Invocation 33284 (0X8204)  
(OS) DosMonReg Pre\_Invocation 00516 (0X0204)  
(OS) DosMonWrite Post\_Invocation 33285 (0X8205)  
(OS) DosMonWrite Pre\_Invocation 00517 (0X0205)

## MONCALLS Major Code: 0X0010 Minor Code: 513 (0X0201)

Description	(OS) DosMonClose Pre_Invocation
Tracepoint	Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.
Minor Code	513 (0X0201)
Trace Groups	TSK
Trace Types	PRE
Traced Parameters	Handle = %w

## MONCALLS Major Code: 0X0010 Minor Code: 514 (0X0202)

<b><u>Description</u></b>	(OS) DosMonOpen Pre_Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.
<b><u>Minor Code</u></b>	514 (0X0202)
<b><u>Trace Groups</u></b>	TSK
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Device name: %s

-----

## MONCALLS Major Code: 0X0010 Minor Code: 515 (0X0203)

<b><u>Description</u></b>	(OS) DosMonRead Pre_Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @monio.c in MONCALLS.
<b><u>Minor Code</u></b>	515 (0X0203)
<b><u>Trace Groups</u></b>	TSK
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Input buffer address = %a Wait flag = %w Data buffer address = %a Byte count = %w

-----

## MONCALLS Major Code: 0X0010 Minor Code: 516 (0X0204)

<b><u>Description</u></b>	(OS) DosMonReg Pre_Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.
<b><u>Minor Code</u></b>	516 (0X0204)

**Trace Groups**

TSK

**Trace Types**

PRE

**Traced Parameters**

Handle = %w

Input buffer address = %a Output buffer address = %a

Position flag = %w Index = %w

Input buffer length = %w Output buffer length = %w

-----

## MONCALLS Major Code: 0X0010 Minor Code: 517 (0X0205)

**Description**

(OS) DosMonWrite Pre\_Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @monio.c in MONCALLS.

**Minor Code**

517 (0X0205)

**Trace Groups**

TSK

**Trace Types**

PRE

**Traced Parameters**

Output buffer address = %a Data buffer address = %a

Byte count = %w Data = %r %b

-----

## MONCALLS Major Code: 0X0010 Minor Code: 33281 (0X8201)

**Description**

(OS) DosMonClose Post\_Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

**Minor Code**

33281 (0X8201)

**Trace Groups**

TSK

**Trace Types**



POST

**Traced Parameters**

Return code = %w

-----

## MONCALLS Major Code: 0X0010 Minor Code: 33282 (0X8202)

**Description**

(OS) DosMonOpen Post\_Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.

**Minor Code**

33282 (0X8202)

**Trace Groups**

TSK

**Trace Types**

API

**Traced Parameters**

Return code = %w New handle = %w

-----

## MONCALLS Major Code: 0X0010 Minor Code: 33283 (0X8203)

**Description**

(OS) DosMonRead Post\_Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @monio.c in MONCALLS.

**Minor Code**

33283 (0X8203)

**Trace Groups**

TSK

**Trace Types**

POST

**Traced Parameters**

Return code = %w Byte count = %w Data = %r %b

-----

# MONCALLS Major Code: 0X0010 Minor Code: 33284 (0X8204)

Description	(OS) DosMonReg Post_Invocation
Tracepoint	Source line defined dynamic tracepoint: @moncalls.c in MONCALLS.
Minor Code	33284 (0X8204)
Trace Groups	TSK
Trace Types	POST
Traced Parameters	Return code = %w Input buffer length = %w Device driver length = %w Output buffer length = %w Device driver length = %w

# MONCALLS Major Code: 0X0010 Minor Code: 33285 (0X8205)

Description	(OS) DosMonWrite Post_Invocation
Tracepoint	Source line defined dynamic tracepoint: @monio.c in MONCALLS.
Minor Code	33285 (0X8205)
Trace Groups	TSK
Trace Types	POST
Traced Parameters	Return code = %w

## OS2CHAR.DLL Trace Events

The tracepoints for the OS2CHAR.DLL major code are identified in the following tables. These tracepoints are dynamic tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for OS2CHAR Major Code: 0X0018, sorted by minor code.  
Trace events for OS2CHAR Major Code: 0X0018 ,sorted by tracepoint.

---

## Trace Events for OS2CHAR Major Code: 0X0018, Sorted by Minor Code

00001 (0X0001) (OS) VioAssociate Pre-Invocation  
00002 (0X0002) (OS) VioCreateLogFont Pre-Invocation  
00003 (0X0003) (OS) VioCreatePS Pre-Invocation  
00004 (0X0004) (OS) VioDeleteSetID Pre-Invocation  
00005 (0X0005) (OS) VioDeRegister Pre-Invocation  
00006 (0X0006) (OS) VioDestroyPS Pre-Invocation  
00007 (0X0007) (OS) VioEndPopUp Pre-Invocation  
00008 (0X0008) (OS) VioGetAnsi Pre-Invocation  
00009 (0X0009) (OS) VioGetBuf Pre-Invocation  
00010 (0X000A) (OS) VioGetConfig Pre-Invocation  
00011 (0X000B) (OS) VioGetCp Pre-Invocation  
00012 (0X000C) (OS) VioGetCurPos Pre-Invocation  
00013 (0X000D) (OS) VioGetCurType Pre-Invocation  
00014 (0X000E) (OS) VioGetDeviceCellSize Pre-Invocation  
00015 (0X000F) (OS) VioGetFont Pre-Invocation  
00016 (0X0010) (OS) VioGetMode Pre-Invocation  
00017 (0X0011) (OS) VioGetOrg Pre-Invocation  
00018 (0X0012) (OS) VioGetPhysBuf Pre-Invocation  
00019 (0X0013) (OS) VioGetPSAddress Pre-Invocation  
00020 (0X0014) (OS) VioGetState Pre-Invocation  
00021 (0X0015) (OS) VioGlobalReg Pre-Invocation  
00022 (0X0016) (OS) VioModeUndo Pre-Invocation  
00023 (0X0017) (OS) VioModeWait Pre-Invocation  
00024 (0X0018) (OS) VioPopUp Pre-Invocation  
00025 (0X0019) (OS) VioPrtSc Pre-Invocation  
00026 (0X001A) (OS) VioPrtScToggle Pre-Invocation  
00027 (0X001B) (OS) VioQueryConsole Pre-Invocation  
00028 (0X001C) (OS) VioQueryFonts Pre-Invocation  
00029 (0X001D) (OS) VioQuerySetIDs Pre-Invocation  
00030 (0X001E) (OS) VioReadCellStr Pre-Invocation  
00031 (0X001F) (OS) VioReadCharStr Pre-Invocation  
00032 (0X0020) (OS) VioRegister Pre-Invocation  
00033 (0X0021) (OS) VioSavRedrawUndo Pre-Invocation  
00034 (0X0022) (OS) VioSavRedrawWait Pre-Invocation  
00035 (0X0023) (OS) VioScrLock Pre-Invocation  
00036 (0X0024) (OS) VioScrollDn Pre-Invocation  
00037 (0X0025) (OS) VioScrollLf Pre-Invocation  
00038 (0X0026) (OS) VioScrollRt Pre-Invocation  
00039 (0X0027) (OS) VioScrollUp Pre-Invocation  
00040 (0X0028) (OS) VioScrUnLock Pre-Invocation  
00041 (0X0029) (OS) VioSetAnsi Pre-Invocation  
00042 (0X002A) (OS) VioSetCp Pre-Invocation  
00043 (0X002B) (OS) VioSetCurPos Pre-Invocation  
00044 (0X002C) (OS) VioSetCurType Pre-Invocation  
00045 (0X002D) (OS) VioSetDeviceCellSize Pre-Invocation  
00046 (0X002E) (OS) VioSetFont Pre-Invocation  
00047 (0X002F) (OS) VioSetMode Pre-Invocation  
00048 (0X0030) (OS) VioSetOrg Pre-Invocation  
00049 (0X0031) (OS) VioSetState Pre-Invocation  
00050 (0X0032) (OS) VioShieldInit Pre-Invocation  
00051 (0X0033) (OS) VioShieldTerm Pre-Invocation  
00052 (0X0034) (OS) VioShowBuf Pre-Invocation

00053 (0X0035) (OS) VioShowPS Pre-Invocation  
00054 (0X0036) (OS) VioWrtCellStr Pre-Invocation  
00055 (0X0037) (OS) VioWrtCharStr Pre-Invocation  
00056 (0X0038) (OS) VioWrtCharStrAtt Pre-Invocation  
00057 (0X0039) (OS) VioWrtNAttr Pre-Invocation  
00058 (0X003A) (OS) VioWrtNCell Pre-Invocation  
00059 (0X003B) (OS) VioWrtNChar Pre-Invocation  
00060 (0X003C) (OS) VioWrtTTY Pre-Invocation  
00257 (0X0101) KbdCharIn Pre-Invocation  
00258 (0X0102) KbdClose Pre-Invocation  
00259 (0X0103) KbdGetHWID Pre-Invocation  
00260 (0X0104) KbdDeRegister Pre-Invocation  
00261 (0X0105) KbdFlushBuffer Pre-Invocation  
00262 (0X0106) KbdFreeFocus Pre-Invocation  
00263 (0X0107) KbdGetCP Pre-Invocation  
00264 (0X0108) KbdGetFocus Pre-Invocation  
00265 (0X0109) KbdGetStatus Pre-Invocation  
00266 (0X010A) KbdOpen Pre-Invocation  
00267 (0X010B) KbdPeek Pre-Invocation  
00268 (0X010C) KbdRegister Pre-Invocation  
00269 (0X010D) KbdSetCP Pre-Invocation  
00270 (0X010E) KbdSetCustXT Pre-Invocation  
00271 (0X010F) KbdSetFgnd Pre-Invocation  
00272 (0X0110) KbdSetStatus Pre-Invocation  
00273 (0X0111) KbdShellInit Pre-Invocation  
00274 (0X0112) KbdStringIn Pre-Invocation  
00275 (0X0113) KbdSynch Pre-Invocation  
00276 (0X0114) KbdXlate Pre-Invocation  
00513 (0X0201) (MOU) MouClose Pre-Invocation  
00514 (0X0202) (MOU) MouDeRegister Pre-Invocation  
00515 (0X0203) (MOU) MouDrawPtr Pre-Invocation  
00516 (0X0204) (MOU) MouFlushQue Pre-Invocation  
00517 (0X0205) (MOU) MouGetDevStatus Pre-Invocation  
00518 (0X0206) (MOU) MouGetEventMask Pre-Invocation  
00519 (0X0207) (MOU) MouGetNumButtons Pre-Invocation  
00520 (0X0208) (MOU) MouGetNumMickeyKeys Pre-Invocation  
00521 (0X0209) (MOU) MouGetNumQueEl Pre-Invocation  
00522 (0X020A) (MOU) MouGetPtrPos Pre-Invocation  
00523 (0X020B) (MOU) MouGetPtrShape Pre-Invocation  
00524 (0X020C) (MOU) MouGetScaleFact Pre-Invocation  
00525 (0X020D) (MOU) MouGetThreshold Pre-Invocation  
00526 (0X020E) (MOU) MouInitReal Pre-Invocation  
00527 (0X020F) (MOU) MouOpen Pre-Invocation  
00528 (0X0210) (MOU) MouReadEventQue Pre-Invocation  
00529 (0X0211) (MOU) MouRegister Pre-Invocation  
00530 (0X0212) (MOU) MouRemovePtr Pre-Invocation  
00531 (0X0213) (MOU) MouSetDevStatus Pre-Invocation  
00532 (0X0214) (MOU) MouSetEventMask Pre-Invocation  
00533 (0X0215) (MOU) MouSetPtrPos Pre-Invocation  
00534 (0X0216) (MOU) MouSetPtrShape Pre-Invocation  
00535 (0X0217) (MOU) MouSetScaleFact Pre-Invocation  
00536 (0X0218) (MOU) MouSetThreshold Pre-Invocation  
00537 (0X0219) (MOU) MouShellInit Pre-Invocation  
00538 (0X021A) (MOU) MouSynch Pre-Invocation  
32769 (0X8001) (OS) VioAssociate Post-Invocation  
32770 (0X8002) (OS) VioCreateLogFont Post-Invocation  
32771 (0X8003) (OS) VioCreatePS Post-Invocation  
32772 (0X8004) (OS) VioDeleteSetID Post-Invocation  
32773 (0X8005) (OS) VioDeRegister Post-Invocation  
32774 (0X8006) (OS) VioDestroyPS Post-Invocation  
32775 (0X8007) (OS) VioEndPopUp Post-Invocation  
32776 (0X8008) (OS) VioGetAnsi Post-Invocation  
32777 (0X8009) (OS) VioGetBuf Post-Invocation  
32778 (0X800A) (OS) VioGetConfig Post-Invocation  
32779 (0X800B) (OS) VioGetCp Post-Invocation  
32780 (0X800C) (OS) VioGetCurPos Post-Invocation  
32781 (0X800D) (OS) VioGetCurType Post-Invocation  
32782 (0X800E) (OS) VioGetDeviceCellSize Post-Invocation  
32783 (0X800F) (OS) VioGetFont Post-Invocation  
32784 (0X8010) (OS) VioGetMode Post-Invocation  
32785 (0X8011) (OS) VioGetOrg Post-Invocation  
32786 (0X8012) (OS) VioGetPhysBuf Post-Invocation  
32787 (0X8013) (OS) VioGetPSAddress Post-Invocation

32788 (0X8014) (OS) VioGetState Post-Invocation  
32789 (0X8015) (OS) VioGlobalReg Post-Invocation  
32790 (0X8016) (OS) VioModeUndo Post-Invocation  
32791 (0X8017) (OS) VioModeWait Post-Invocation  
32792 (0X8018) (OS) VioPopUp Post-Invocation  
32793 (0X8019) (OS) VioPrtSc Post-Invocation  
32794 (0X801A) (OS) VioPrtScToggle Post-Invocation  
32795 (0X801B) (OS) VioQueryConsole Post-Invocation  
32796 (0X801C) (OS) VioQueryFonts Post-Invocation  
32797 (0X801D) (OS) VioQuerySetIDs Post-Invocation  
32798 (0X801E) (OS) VioReadCellStr Post-Invocation  
32799 (0X801F) (OS) VioReadCharStr Post-Invocation  
32800 (0X8020) (OS) VioRegister Post-Invocation  
32801 (0X8021) (OS) VioSavRedrawUndo Post-Invocation  
32802 (0X8022) (OS) VioSavRedrawWait Post-Invocation  
32803 (0X8023) (OS) VioScrLock Post-Invocation  
32804 (0X8024) (OS) VioScrollDn Post-Invocation  
32805 (0X8025) (OS) VioScrollLf Post-Invocation  
32806 (0X8026) (OS) VioScrollRt Post-Invocation  
32807 (0X8027) (OS) VioScrollUp Post-Invocation  
32808 (0X8028) (OS) VioScrUnLock Post-Invocation  
32809 (0X8029) (OS) VioSetAnsi Post-Invocation  
32810 (0X802A) (OS) VioSetCp Post-Invocation  
32811 (0X802B) (OS) VioSetCurPos Post-Invocation  
32812 (0X802C) (OS) VioSetCurType Post-Invocation  
32813 (0X802D) (OS) VioSetDeviceCellSize Post-Invocation  
32814 (0X802E) (OS) VioSetFont Post-Invocation  
32815 (0X802F) (OS) VioSetMode Post-Invocation  
32816 (0X8030) (OS) VioSetOrg Post-Invocation  
32817 (0X8031) (OS) VioSetState Post-Invocation  
32818 (0X8032) (OS) VioShieldInit Post-Invocation  
32819 (0X8033) (OS) VioShieldTerm Post-Invocation  
32820 (0X8034) (OS) VioShowBuf Post-Invocation  
32821 (0X8035) (OS) VioShowPS Post-Invocation  
32822 (0X8036) (OS) VioWrtCellStr Post-Invocation  
32823 (0X8037) (OS) VioWrtCharStr Post-Invocation  
32824 (0X8038) (OS) VioWrtCharStrAtt Post-Invocation  
32825 (0X8039) (OS) VioWrtNAttr Post-Invocation  
32826 (0X803A) (OS) VioWrtNCell Post-Invocation  
32827 (0X803B) (OS) VioWrtNChar Post-Invocation  
32828 (0X803C) (OS) VioWrtTTY Post-Invocation  
33025 (0X8101) KbdCharIn Post-Invocation  
33026 (0X8102) KbdClose Post-Invocation  
33027 (0X8103) KbdGetHWID Post-Invocation  
33028 (0X8104) KbdDeRegister Post-Invocation  
33029 (0X8105) KbdFlushBuffer Post-Invocation  
33030 (0X8106) KbdFreeFocus Post-Invocation  
33031 (0X8107) KbdGetCP Post-Invocation  
33032 (0X8108) KbdGetFocus Post-Invocation  
33033 (0X8109) KbdGetStatus Post-Invocation  
33034 (0X810A) KbdOpen Post-Invocation  
33035 (0X810B) KbdPeek Post-Invocation  
33036 (0X810C) KbdRegister Post-Invocation  
33037 (0X810D) KbdSetCP Post-Invocation  
33038 (0X810E) KbdSetCustXT Post-Invocation  
33039 (0X810F) KbdSetFgnd Post-Invocation  
33040 (0X8110) KbdSetStatus Post-Invocation  
33041 (0X8111) KbdShellInit Post-Invocation  
33042 (0X8112) KbdStringIn Post-Invocation  
33043 (0X8113) KbdSynch Post-Invocation  
33044 (0X8114) KbdXlate Post-Invocation  
33281 (0X8201) (MOU) MouClose Post-Invocation  
33282 (0X8202) (MOU) MouDeRegister Post-Invocation  
33283 (0X8203) (MOU) MouDrawPtr Post-Invocation  
33284 (0X8204) (MOU) MouFlushQue Post-Invocation  
33285 (0X8205) (MOU) MouGetDevStatus Post-Invocation  
33286 (0X8206) (MOU) MouGetEventMask Post-Invocation  
33287 (0X8207) (MOU) MouGetNumButtons Post-Invocation  
33288 (0X8208) (MOU) MouGetNumMickeyKeys Post-Invocation  
33289 (0X8209) (MOU) MouGetNumQueEI Post-Invocation  
33290 (0X820A) (MOU) MouGetPtrPos Post-Invocation  
33291 (0X820B) (MOU) MouGetPtrShape Post-Invocation  
33292 (0X820C) (MOU) MouGetScaleFact Post-Invocation

33293 (0X820D) (MOU) MouGetThreshold Post-Invocation  
33294 (0X820E) (MOU) MouInitReal Post-Invocation  
33295 (0X820F) (MOU) MouOpen Post-Invocation  
33296 (0X8210) (MOU) MouReadEventQue Post-Invocation  
33297 (0X8211) (MOU) MouRegister Post-Invocation  
33298 (0X8212) (MOU) MouRemovePtr Post-Invocation  
33299 (0X8213) (MOU) MouSetDevStatus Post-Invocation  
33300 (0X8214) (MOU) MouSetEventMask Post-Invocation  
33301 (0X8215) (MOU) MouSetPtrPos Post-Invocation  
33302 (0X8216) (MOU) MouSetPtrShape Post-Invocation  
33303 (0X8217) (MOU) MouSetScaleFact Post-Invocation  
33304 (0X8218) (MOU) MouSetThreshold Post-Invocation  
33305 (0X8219) (MOU) MouShellInit Post-Invocation  
33306 (0X821A) (MOU) MouSynch Post-Invocation

---

## Trace Events for OS2CHAR Major Code: 0X0018, Sorted by Tracepoint

KBDCHARIN 00257 (0X0101)  
KBDCHARIN 33025 (0X8101)  
KBDCLOSE 00258 (0X0102)  
KBDCLOSE 33026 (0X8102)  
KBDDEREGISTER 00260 (0X0104)  
KBDDEREGISTER 33028 (0X8104)  
KBDFLUSHBUFFER 00261 (0X0105)  
KBDFLUSHBUFFER 33029 (0X8105)  
KBDFREEFOCUS 00262 (0X0106)  
KBDFREEFOCUS 33030 (0X8106)  
KBDGETCP 00263 (0X0107)  
KBDGETCP 33031 (0X8107)  
KBDGETFOCUS 00264 (0X0108)  
KBDGETFOCUS 33032 (0X8108)  
KBDGETHWID 00259 (0X0103)  
KBDGETHWID 33027 (0X8103)  
KBDGETSTATUS 00265 (0X0109)  
KBDGETSTATUS 33033 (0X8109)  
KBDOPEN 00266 (0X010A)  
KBDOPEN 33034 (0X810A)  
KBDPEEK 00267 (0X010B)  
KBDPEEK 33035 (0X810B)  
KBDREGISTER 00268 (0X010C)  
KBDREGISTER 33036 (0X810C)  
KBDSETCP 00269 (0X010D)  
KBDSETCP 33037 (0X810D)  
KBDSETCUSTXT 00270 (0X010E)  
KBDSETCUSTXT 33038 (0X810E)  
KBDSETFGND 00271 (0X010F)  
KBDSETFGND 33039 (0X810F)  
KBDSETSTATUS 00272 (0X0110)  
KBDSETSTATUS 33040 (0X8110)  
KBDSHELLINIT 00273 (0X0111)  
KBDSHELLINIT 33041 (0X8111)  
KBDSTRINGIN 00274 (0X0112)  
KBDSTRINGIN 33042 (0X8112)  
KBDSYNCH 00275 (0X0113)  
KBDSYNCH 33043 (0X8113)  
KBDXLATE 00276 (0X0114)  
KBDXLATE 33044 (0X8114)  
MOUCLOSE 00513 (0X0201)  
MOUCLOSE 33281 (0X8201)  
MOUDEREGISTER 00514 (0X0202)  
MOUDEREGISTER 33282 (0X8202)  
MOUDRAWPTR 00515 (0X0203)  
MOUDRAWPTR 33283 (0X8203)

MOUFLUSHQUE 00516 (0X0204)  
MOUFLUSHQUE 33284 (0X8204)  
MOUGETDEVSTATUS 00517 (0X0205)  
MOUGETDEVSTATUS 33285 (0X8205)  
MOUGETEVENTMASK 00518 (0X0206)  
MOUGETEVENTMASK 33286 (0X8206)  
MOUGETNUMBUTTONS 00519 (0X0207)  
MOUGETNUMBUTTONS 33287 (0X8207)  
MOUGETNUMMICKEYS 00520 (0X0208)  
MOUGETNUMMICKEYS 33288 (0X8208)  
MOUGETNUMQUEEL 00521 (0X0209)  
MOUGETNUMQUEEL 33289 (0X8209)  
MOUGETPTRPOS 00522 (0X020A)  
MOUGETPTRPOS 33290 (0X820A)  
MOUGETPTRSHAPE 00523 (0X020B)  
MOUGETPTRSHAPE 33291 (0X820B)  
MOUGETSCALEFACT 00524 (0X020C)  
MOUGETSCALEFACT 33292 (0X820C)  
MOUGETTHRESHOLD 00525 (0X020D)  
MOUGETTHRESHOLD 33293 (0X820D)  
MOUINITREAL 00526 (0X020E)  
MOUINITREAL 33294 (0X820E)  
MOUOPEN 00527 (0X020F)  
MOUOPEN 33295 (0X820F)  
MOUREADEVENTQUE 00528 (0X0210)  
MOUREADEVENTQUE 33296 (0X8210)  
MOUREGISTER 00529 (0X0211)  
MOUREGISTER 33297 (0X8211)  
MOUREMOVEPTR 00530 (0X0212)  
MOUREMOVEPTR 33298 (0X8212)  
MOUSETDEVSTATUS 00531 (0X0213)  
MOUSETDEVSTATUS 33299 (0X8213)  
MOUSETEVENTMASK 00532 (0X0214)  
MOUSETEVENTMASK 33300 (0X8214)  
MOUSETPTRPOS 00533 (0X0215)  
MOUSETPTRPOS 33301 (0X8215)  
MOUSETPTRSHAPE 00534 (0X0216)  
MOUSETPTRSHAPE 33302 (0X8216)  
MOUSETSCALEFACT 00535 (0X0217)  
MOUSETSCALEFACT 33303 (0X8217)  
MOUSETTHRESHOLD 00536 (0X0218)  
MOUSETTHRESHOLD 33304 (0X8218)  
MOUSHELLINIT 00537 (0X0219)  
MOUSHELLINIT 33305 (0X8219)  
MOUSYNCH 00538 (0X021A)  
MOUSYNCH 33306 (0X821A)  
VIOASSOCIATE 00001 (0X0001)  
VIOASSOCIATE\_POSTDT 32769 (0X8001)  
VIOCREATELOGFONT 00002 (0X0002)  
VIOCREATELOGFONT\_POSTDT 32770 (0X8002)  
VIOCREATEPS 00003 (0X0003)  
VIOCREATEPS\_POSTDT 32771 (0X8003)  
VIODELETESETID 00004 (0X0004)  
VIODELETESETID\_POSTDT 32772 (0X8004)  
VIODEREGISTER 00005 (0X0005)  
VIODEREGISTER\_POSTDT 32773 (0X8005)  
VIODESTROYPS 00006 (0X0006)  
VIODESTROYPS\_POSTDT 32774 (0X8006)  
VIOENDPOPUP 00007 (0X0007)  
VIOENDPOPUP\_POSTDT 32775 (0X8007)  
VIOGETANSI 00008 (0X0008)  
VIOGETANSI\_POSTDT 32776 (0X8008)  
VIOGETBUF 00009 (0X0009)  
VIOGETBUF\_POSTDT 32777 (0X8009)  
VIOGETCONFIG 00010 (0X000A)  
VIOGETCONFIG\_POSTDT 32778 (0X800A)  
VIOGETCP 00011 (0X000B)  
VIOGETCP\_POSTDT 32779 (0X800B)  
VIOGETCURPOS 00012 (0X000C)  
VIOGETCURPOS\_POSTDT 32780 (0X800C)  
VIOGETCURTYPE 00013 (0X000D)  
VIOGETCURTYPE\_POSTDT 32781 (0X800D)  
VIOGETDEVICECELLSIZE 00014 (0X000E)

VIOGETDEVICECELLSIZE\_POSTDT 32782 (0X800E)  
VIOGETFONT 00015 (0X000F)  
VIOGETFONT\_POSTDT 32783 (0X800F)  
VIOGETMODE 00016 (0X0010)  
VIOGETMODE\_POSTDT 32784 (0X8010)  
VIOGETORG 00017 (0X0011)  
VIOGETORG\_POSTDT 32785 (0X8011)  
VIOGETPHYSBUF 00018 (0X0012)  
VIOGETPHYSBUF\_POSTDT 32786 (0X8012)  
VIOGETPSADDRESS 00019 (0X0013)  
VIOGETPSADDRESS\_POSTDT 32787 (0X8013)  
VIOGETSTATE 00020 (0X0014)  
VIOGETSTATE\_POSTDT 32788 (0X8014)  
VIOGLOBALREG 00021 (0X0015)  
VIOGLOBALREG\_POSTDT 32789 (0X8015)  
VIOMODEUNDO 00022 (0X0016)  
VIOMODEUNDO\_POSTDT 32790 (0X8016)  
VIOMODEWAIT 00023 (0X0017)  
VIOMODEWAIT\_POSTDT 32791 (0X8017)  
VIOPOPUP 00024 (0X0018)  
VIOPOPUP\_POSTDT 32792 (0X8018)  
VIOVRTSC 00025 (0X0019)  
VIOVRTSCTOGGLE 00026 (0X001A)  
VIOVRTSCTOGGLE\_POSTDT 32794 (0X801A)  
VIOVRTSC\_POSTDT 32793 (0X8019)  
VIOQUERYCONSOLE 00027 (0X001B)  
VIOQUERYCONSOLE\_POSTDT 32795 (0X801B)  
VIOQUERYFONTS 00028 (0X001C)  
VIOQUERYFONTS\_POSTDT 32796 (0X801C)  
VIOQUERYSETIDS 00029 (0X001D)  
VIOQUERYSETIDS\_POSTDT 32797 (0X801D)  
VIOREADCELLSTR 00030 (0X001E)  
VIOREADCELLSTR\_POSTDT 32798 (0X801E)  
VIOREADCHARSTR 00031 (0X001F)  
VIOREADCHARSTR\_POSTDT 32799 (0X801F)  
VIOREGISTER 00032 (0X0020)  
VIOREGISTER\_POSTDT 32800 (0X8020)  
VIOAVREDRAWUNDO 00033 (0X0021)  
VIOAVREDRAWUNDO\_POSTDT 32801 (0X8021)  
VIOAVREDRAWWAIT 00034 (0X0022)  
VIOAVREDRAWWAIT\_POSTDT 32802 (0X8022)  
VIOSCRLOCK 00035 (0X0023)  
VIOSCRLOCK\_POSTDT 32803 (0X8023)  
VIOSCROLLDN 00036 (0X0024)  
VIOSCROLLDN\_POSTDT 32804 (0X8024)  
VIOSCROLLLF 00037 (0X0025)  
VIOSCROLLLF\_POSTDT 32805 (0X8025)  
VIOSCROLLRT 00038 (0X0026)  
VIOSCROLLRT\_POSTDT 32806 (0X8026)  
VIOSCROLLUP 00039 (0X0027)  
VIOSCROLLUP\_POSTDT 32807 (0X8027)  
VIOSCRUNLOCK 00040 (0X0028)  
VIOSCRUNLOCK\_POSTDT 32808 (0X8028)  
VIOSETANSI 00041 (0X0029)  
VIOSETANSI\_POSTDT 32809 (0X8029)  
VIOSETCP 00042 (0X002A)  
VIOSETCP\_POSTDT 32810 (0X802A)  
VIOSETCURPOS 00043 (0X002B)  
VIOSETCURPOS\_POSTDT 32811 (0X802B)  
VIOSETCURTYPE 00044 (0X002C)  
VIOSETCURTYPE\_POSTDT 32812 (0X802C)  
VIOSETDEVICECELLSIZE 00045 (0X002D)  
VIOSETDEVICECELLSIZE\_POSTDT 32813 (0X802D)  
VIOSETFONT 00046 (0X002E)  
VIOSETFONT\_POSTDT 32814 (0X802E)  
VIOSETMODE 00047 (0X002F)  
VIOSETMODE\_POSTDT 32815 (0X802F)  
VIOSETORG 00048 (0X0030)  
VIOSETORG\_POSTDT 32816 (0X8030)  
VIOSETSTATE 00049 (0X0031)  
VIOSETSTATE\_POSTDT 32817 (0X8031)  
VIOSHIELDINIT 00050 (0X0032)  
VIOSHIELDINIT\_POSTDT 32818 (0X8032)



VIOSHIELDTERM 00051 (0X0033)  
VIOSHIELDTERM\_POSTDT 32819 (0X8033)  
VIOSHOWBUF 00052 (0X0034)  
VIOSHOWBUF\_POSTDT 32820 (0X8034)  
VIOSHOWPS 00053 (0X0035)  
VIOSHOWPS\_POSTDT 32821 (0X8035)  
VIOVRTCELLSTR 00054 (0X0036)  
VIOVRTCELLSTR\_POSTDT 32822 (0X8036)  
VIOVRTCHARSTR 00055 (0X0037)  
VIOVRTCHARSTRATT 00056 (0X0038)  
VIOVRTCHARSTRATT\_POSTDT 32824 (0X8038)  
VIOVRTCHARSTR\_POSTDT 32823 (0X8037)  
VIOVRTNATTR 00057 (0X0039)  
VIOVRTNATTR\_POSTDT 32825 (0X8039)  
VIOVRTNCELL 00058 (0X003A)  
VIOVRTNCELL\_POSTDT 32826 (0X803A)  
VIOVRTNCHAR 00059 (0X003B)  
VIOVRTNCHAR\_POSTDT 32827 (0X803B)  
VIOVRTTTY 00060 (0X003C)  
VIOVRTTTY\_POSTDT 32828 (0X803C)

## OS2CHAR Major Code: 0X0018 Minor Code: 1 (0X0001)

### Description

(OS) VioAssociate Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOASSOCIATE

### Minor Code

1 (0X0001)

### Trace Groups

VIO

### Trace Types

PRE

### Traced Parameters

PS Handle = %w

Device Context Handle = %d

## OS2CHAR Major Code: 0X0018 Minor Code: 2 (0X0002)

### Description

(OS) VioCreateLogFont Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATELOGFONT

### Minor Code

2 (0X0002)

### Trace Groups

VIO

**Trace Types**

PRE

**Traced Parameters**

- PS Handle = %w
- Logical Font Name = %s
- Local Identifier = %d
- FATTRS - usRecordLength = %w
- fsSelection = %w
- lMatch = %d
- szFacename = %s
- idRegistry = %w
- usCodePage = %w
- lMaxBaseLineExt = %d
- lAveCharWidth = %d
- fsType = %w
- fsFontUse = %w

OS2CHAR Major Code: 0X0018 Minor Code: 3 (0X0003)

**Description**

(OS) VioCreatePS Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATEPS

**Minor Code**

3 (0X0003)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

- Reserved = %w
- Attribute Size = %w
- Format = %w
- Width = %w
- Depth = %w
- PS Handle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 4 (0X0004)

<u>Description</u>	(OS) VioDeleteSetID Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIODELETESETID
<u>Minor Code</u>	4 (0X0004)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	PS Handle = %w
	Local Identifier = %d

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 5 (0X0005)

<u>Description</u>	(OS) VioDeRegister Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIODEREGISTER
<u>Minor Code</u>	5 (0X0005)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	None

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 6 (0X0006)

<u>Description</u>	(OS) VioDestroyPS Pre-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: OS2CHAR.VIODESTROYPS

**Minor Code**

6 (0X0006)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

PS Handle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 7 (0X0007)

**Description**

(OS) VioEndPopUp Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOENDPOPUP

**Minor Code**

7 (0X0007)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 8 (0X0008)

**Description**

(OS) VioGetAnsi Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETANSI

**Minor Code**

8 (0X0008)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Indicator	= %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 9 (0X0009)

### Description

(OS) VioGetBuf Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETBUF

### Minor Code

9 (0X0009)

### Trace Groups

VIO

### Trace Types

PRE

### Traced Parameters

Handle	= %w
Length	= %w
LVBptr	= %d

---

## OS2CHAR Major Code: 0X0018 Minor Code: 10 (0X000A)

### Description

(OS) VioGetConfig Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCONFIG

### Minor Code

10 (0X000A)

### Trace Groups

VIO

### Trace Types

PRE

### Traced Parameters

Handle	= %w
Config Data	= %r%w
Reserved	= %w

# OS2CHAR Major Code: 0X0018 Minor Code: 11 (0X000B)

<u>Description</u>	(OS) VioGetCp Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCP
<u>Minor Code</u>	11 (0X000B)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
Handle	= %w
Code Page Id	= %w
Reserved	= %w

# OS2CHAR Major Code: 0X0018 Minor Code: 12 (0X000C)

<u>Description</u>	(OS) VioGetCurPos Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURPOS
<u>Minor Code</u>	12 (0X000C)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
Handle	= %w
Column	= %w
Row	= %w

# OS2CHAR Major Code: 0X0018 Minor Code: 13 (0X000D)

<b>Description</b>	(OS) VioGetCurType Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURTYPE
<b>Minor Code</b>	13 (0X000D)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	
Handle	= %w
Cursor Data	= %w%w%w%w%w

## OS2CHAR Major Code: 0X0018 Minor Code: 14 (0X000E)

<b>Description</b>	(OS) VioGetDeviceCellSize Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETDEVICECELLSIZE
<b>Minor Code</b>	14 (0X000E)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	
PS Handle	= %w
Width	= %w
Height	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 15 (0X000F)

<b>Description</b>	(OS) VioGetFont Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETFONT
<b>Minor Code</b>	15 (0X000F)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w  
Request Block = %r%w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 16 (0X0010)

**Description**

(OS) VioGetMode Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETMODE

**Minor Code**

16 (0X0010)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w  
ModeData = %r%w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 17 (0X0011)

**Description**

(OS) VioGetOrg Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETORG

**Minor Code**

17 (0X0011)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**



PS Handle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 18 (0X0012)

### Description

(OS) VioGetPhysBuf Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPHYSBUF

### Minor Code

18 (0X0012)

### Trace Groups

VIO

### Trace Types

PRE

### Traced Parameters

Reserved = %w

DisplayBuf = %d%d%w%w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 19 (0X0013)

### Description

(OS) VioGetPSAddress Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPSADDRESS

### Minor Code

19 (0X0013)

### Trace Groups

VIO

### Trace Types

PRE

### Traced Parameters

PS Handle = %w

PS Address = %d

Reserved = %d

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 20 (0X0014)

<u>Description</u>	(OS) VioGetState Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETSTATE
<u>Minor Code</u>	20 (0X0014)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %w
	Request Block = %r%w

## OS2CHAR Major Code: 0X0018 Minor Code: 21 (0X0015)

<u>Description</u>	(OS) VioGlobalReg Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGLOBALREG
<u>Minor Code</u>	21 (0X0015)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Reserved = %d
	Function Mask 2 = %d
	Function Mask 1 = %d
	Entry Point Name = %s
	Module Name = %s

## OS2CHAR Major Code: 0X0018 Minor Code: 22 (0X0016)

<u>Description</u>
--------------------

(OS) VioModeUndo Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEUNDO

**Minor Code**

22 (0X0016)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Reserved = %w

Kill Indicator = %w

Owner Indicator = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 23 (0X0017)

**Description**

(OS) VioModeWait Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEWAIT

**Minor Code**

23 (0X0017)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Reserved = %w

Notify Type = %w

Request Type = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 24 (0X0018)

**Description**

(OS) VioPopUp Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOPOPUP

**Minor Code**

24 (0X0018)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w

Options = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 25 (0X0019)

**Description**

(OS) VioPrtSc Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSC

**Minor Code**

25 (0X0019)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 26 (0X001A)

**Description**

(OS) VioPrtScToggle Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSCTOGGLE

**Minor Code**

26 (0X001A)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 27 (0X001B)

<u>Description</u>	(OS) VioQueryConsole Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYCONSOLE
<u>Minor Code</u>	27 (0X001B)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Console Structure = %d

-----

OS2CHAR Major Code: 0X0018 Minor Code: 28 (0X001C)

<u>Description</u>	(OS) VioQueryFonts Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYFONTS
<u>Minor Code</u>	28 (0X001C)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	PS Handle = %w
	Options = %d
	Font Face Name = %s
	Fonts Returned = %d
	Metrics Length = %d

-----

OS2CHAR Major Code: 0X0018 Minor Code: 29 (0X001D)

<b><u>Description</u></b>	(OS) VioQuerySetIDs Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYSETIDS
<b><u>Minor Code</u></b>	29 (0X001D)
<b><u>Trace Groups</u></b>	VIO
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	PS Handle = %w
	Count = %d
	Types = %d
	Font Names = %r%b
	Local Identifiers = %d

## OS2CHAR Major Code: 0X0018 Minor Code: 30 (0X001E)

<b><u>Description</u></b>	(OS) VioReadCellStr Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCELLSTR
<b><u>Minor Code</u></b>	30 (0X001E)
<b><u>Trace Groups</u></b>	VIO
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	Handle = %w
	Column = %w
	Row = %w
	Length = %w
	CellStr = %r%b

## OS2CHAR Major Code: 0X0018 Minor Code: 31 (0X001F)

<b>Description</b>	(OS) VioReadCharStr Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCHARSTR
<b>Minor Code</b>	31 (0X001F)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	
Handle	= %w
Column	= %w
Row	= %w
Length	= %w
CharStr	= %r%b

## OS2CHAR Major Code: 0X0018 Minor Code: 32 (0X0020)

<b>Description</b>	(OS) VioRegister Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOREGISTER
<b>Minor Code</b>	32 (0X0020)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	
Function Mask 2	= %d
Function Mask 1	= %d
Entry Point Name	= %s
Module Name	= %s

## OS2CHAR Major Code: 0X0018 Minor Code: 33 (0X0021)

**Description**

(OS) VioSavRedrawUndo Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWUNDO

**Minor Code**

33 (0X0021)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Reserved	= %w
Kill Indicator	= %w
Owner Indicator	= %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 34 (0X0022)

**Description**

(OS) VioSavRedrawWait Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWWAIT

**Minor Code**

34 (0X0022)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Reserved	= %w
Notify Type	= %w
SavRedrawIndic	= %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 35 (0X0023)

**Description**

(OS) VioScrLock Pre-Invocation



<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCRLOCK	
<b><u>Minor Code</u></b>	35 (0X0023)	
<b><u>Trace Groups</u></b>	VIO	
<b><u>Trace Types</u></b>	PRE	
<b><u>Traced Parameters</u></b>		
	Handle	= %w
	Status	= %b
	Wait Flag	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 36 (0X0024)

<b><u>Description</u></b>	(OS) VioScrollDn Pre-Invocation	
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLDN	
<b><u>Minor Code</u></b>	36 (0X0024)	
<b><u>Trace Groups</u></b>	VIO	
<b><u>Trace Types</u></b>	PRE	
<b><u>Traced Parameters</u></b>		
	Handle	= %w
	Cell	= %w
	Lines	= %w
	RightCol	= %w
	BotRow	= %w
	LeftCol	= %w
	TopRow	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 37 (0X0025)

<b><u>Description</u></b>
---------------------------

(OS) VioScrollLf Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLLF

**Minor Code**

37 (0X0025)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Cell	= %w
Lines	= %w
RightCol	= %w
BotRow	= %w
LeftCol	= %w
TopRow	= %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 38 (0X0026)

**Description**

(OS) VioScrollRt Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLRT

**Minor Code**

38 (0X0026)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Cell	= %w
Lines	= %w
RightCol	= %w
BotRow	= %w
LeftCol	= %w
TopRow	= %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 39 (0X0027)

<u>Description</u>	(OS) VioScrollUp Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLUP
<u>Minor Code</u>	39 (0X0027)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %w
	Cell = %w
	Lines = %w
	RightCol = %w
	BotRow = %w
	LeftCol = %w
	TopRow = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 40 (0X0028)

<u>Description</u>	(OS) VioScrUnLock Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCRUNLOCK
<u>Minor Code</u>	40 (0X0028)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Handle = %w

-----

# OS2CHAR Major Code: 0X0018 Minor Code: 41 (0X0029)

Description	(OS) VioSetAnsi Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETANSI
Minor Code	41 (0X0029)
Trace Groups	VIO
Trace Types	PRE
Traced Parameters	
Handle	= %w
Indicator	= %w

# OS2CHAR Major Code: 0X0018 Minor Code: 42 (0X002A)

Description	(OS) VioSetCp Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCP
Minor Code	42 (0X002A)
Trace Groups	VIO
Trace Types	PRE
Traced Parameters	
Handle	= %w
Code Page Id	= %w
Reserved	= %w

# OS2CHAR Major Code: 0X0018 Minor Code: 43 (0X002B)

Description	(OS) VioSetCurPos Pre-Invocation
-------------	----------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURPOS	
<u>Minor Code</u>	43 (0X002B)	
<u>Trace Groups</u>	VIO	
<u>Trace Types</u>	PRE	
<u>Traced Parameters</u>		
	Handle	= %w
	Column	= %w
	Row	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 44 (0X002C)

<u>Description</u>	(OS) VioSetCurType Pre-Invocation	
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURTYPE	
<u>Minor Code</u>	44 (0X002C)	
<u>Trace Groups</u>	VIO	
<u>Trace Types</u>	PRE	
<u>Traced Parameters</u>		
	Handle	= %w
	CursorData	= %w%w%w%w

## OS2CHAR Major Code: 0X0018 Minor Code: 45 (0X002D)

<u>Description</u>	(OS) VioSetDeviceCellSize Pre-Invocation	
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETDEVICECELLSIZE	
<u>Minor Code</u>	45 (0X002D)	
<u>Trace Groups</u>		

VIO

**Trace Types**

PRE

**Traced Parameters**

PS Handle = %w

Width = %w

Height = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 46 (0X002E)

**Description**

(OS) VioSetFont Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETFONT

**Minor Code**

46 (0X002E)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w

Request Block = %r%w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 47 (0X002F)

**Description**

(OS) VioSetMode Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETMODE

**Minor Code**

47 (0X002F)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
ModeData	= %r%w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 48 (0X0030)

### Description

(OS) VioSetOrg Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETORG

### Minor Code

48 (0X0030)

### Trace Groups

VIO

### Trace Types

PRE

### Traced Parameters

PS Handle	= %w
Column	= %w
Row	= %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 49 (0X0031)

### Description

(OS) VioSetState Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETSTATE

### Minor Code

49 (0X0031)

### Trace Groups

VIO

### Trace Types

PRE

### Traced Parameters

Handle	= %w
Request Block	= %r%w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 50 (0X0032)

<b>Description</b>	(OS) VioShieldInit Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDINIT
<b>Minor Code</b>	50 (0X0032)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	None

## OS2CHAR Major Code: 0X0018 Minor Code: 51 (0X0033)

<b>Description</b>	(OS) VioShieldTerm Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDTERM
<b>Minor Code</b>	51 (0X0033)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	None

## OS2CHAR Major Code: 0X0018 Minor Code: 52 (0X0034)

<b>Description</b>	(OS) VioShowBuf Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWBUF
<b>Minor Code</b>	52 (0X0034)
<b>Trace Groups</b>	VIO



**Trace Types**

PRE

**Traced Parameters**

Handle = %w

Length = %w

Offset = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 53 (0X0035)

**Description**

(OS) VioShowPS Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWPS

**Minor Code**

53 (0X0035)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

PS Handle = %w

Cell Offset = %w

Width = %w

Depth = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 54 (0X0036)

**Description**

(OS) VioWrtCellStr Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCELLSTR

**Minor Code**

54 (0X0036)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Column	= %w
Row	= %w
Length	= %w
Cell String	= %r%b

-----

OS2CHAR Major Code: 0X0018 Minor Code: 55 (0X0037)

**Description**

(OS) VioWrtCharStr Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCHARSTR

**Minor Code**

55 (0X0037)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Column	= %w
Row	= %w
Length	= %w
Char String	= %r%b

-----

OS2CHAR Major Code: 0X0018 Minor Code: 56 (0X0038)

**Description**

(OS) VioWrtCharStrAtt Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCHARSTRATT

**Minor Code**

56 (0X0038)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Attr	= %b
Column	= %w
Row	= %w
Length	= %w
CharStr	= %r%b

OS2CHAR Major Code: 0X0018 Minor Code: 57 (0X0039)

**Description**

(OS) VioWrtNAttr Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNATTR

**Minor Code**

57 (0X0039)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Column	= %w
Row	= %w
Times	= %w
Attr	= %b

OS2CHAR Major Code: 0X0018 Minor Code: 58 (0X003A)

**Description**

(OS) VioWrtNCell Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNCELL

**Minor Code**

58 (0X003A)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Column	= %w
Row	= %w
Times	= %w
Cell	= %b%b

-----

OS2CHAR Major Code: 0X0018 Minor Code: 59 (0X003B)

**Description**

(OS) VioWrtNChar Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNCHAR

**Minor Code**

59 (0X003B)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle	= %w
Column	= %w
Row	= %w
Times	= %w
Char	= %b

-----

OS2CHAR Major Code: 0X0018 Minor Code: 60 (0X003C)

**Description**

(OS) VioWrtTTY Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTTTY

**Minor Code**

60 (0X003C)

**Trace Groups**

VIO

**Trace Types**

PRE

**Traced Parameters**

Handle = %w

Length = %w

CharStr = %r%b

---

## OS2CHAR Major Code: 0X0018 Minor Code: 257 (0X0101)

**Description**

KbdCharIn Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDCHARIN

**Minor Code**

257 (0X0101)

**Trace Groups**

KBD

**Trace Types**

PRE

**Traced Parameters**

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

IOWAIT = %W

CharDataRec addr. = %A

---

## OS2CHAR Major Code: 0X0018 Minor Code: 258 (0X0102)

**Description**

KbdClose Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDCLOSE

**Minor Code**

258 (0X0102)

**Trace Groups**

KBD

**Trace Types**

PRE

**Traced Parameters**

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 259 (0X0103)

**Description**

KbdGetHWID Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETHWID

**Minor Code**

259 (0X0103)

**Trace Groups**

KBD

**Trace Types**

PRE

**Traced Parameters**

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

HWIDStruc addr. = %A

HWIDStruc length = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 260 (0X0104)

<b><u>Description</u></b>	KbdDeRegister Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDDEREGISTER
<b><u>Minor Code</u></b>	260 (0X0104)
<b><u>Trace Groups</u></b>	KBD
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Major = %X   Minor = %Y ProcStatus = %B ProcType   = %B SessionID = %W

## OS2CHAR Major Code: 0X0018 Minor Code: 261 (0X0105)

<b><u>Description</u></b>	KbdFlushBuffer Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDFLUSHBUFFER
<b><u>Minor Code</u></b>	261 (0X0105)
<b><u>Trace Groups</u></b>	KBD
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Major = %X   Minor = %Y ProcStatus = %B ProcType   = %B SessionID = %W KbdHandle = %W

## OS2CHAR Major Code: 0X0018 Minor Code: 262 (0X0106)

<u>Description</u>	KbdFreeFocus Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDFREEFOCUS
<u>Minor Code</u>	262 (0X0106)
<u>Trace Groups</u>	KBD
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Major = %X  Minor = %Y ProcStatus = %B ProcType  = %B SessionID = %W KbdHandle = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 263 (0X0107)

<u>Description</u>	KbdGetCP Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETCP
<u>Minor Code</u>	263 (0X0107)
<u>Trace Groups</u>	KBD
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Major = %X  Minor = %Y ProcStatus = %B ProcType  = %B SessionID = %W KbdHandle = %W CodePageID addr. = %A

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 264 (0X0108)



<b><u>Description</u></b>	KbdGetFocus Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETFOCUS
<b><u>Minor Code</u></b>	264 (0X0108)
<b><u>Trace Groups</u></b>	KBD
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>Major = %X   Minor = %Y</p> <p>ProcStatus = %B</p> <p>ProcType   = %B</p> <p>SessionID = %W</p> <p>KbdHandle = %W</p> <p>IOWAIT    = %W</p>

## OS2CHAR Major Code: 0X0018 Minor Code: 265 (0X0109)

<b><u>Description</u></b>	KbdGetStatus Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETSTATUS
<b><u>Minor Code</u></b>	265 (0X0109)
<b><u>Trace Groups</u></b>	KBD
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>Major = %X   Minor = %Y</p> <p>ProcStatus = %B</p> <p>ProcType   = %B</p> <p>SessionID = %W</p> <p>KbdHandle = %W</p> <p>Status addr. = %A</p>

---

## OS2CHAR Major Code: 0X0018 Minor Code: 266 (0X010A)

### Description

KbdOpen Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDOPEN

### Minor Code

266 (0X010A)

### Trace Groups

KBD

### Trace Types

PRE

### Traced Parameters

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle addr. = %A

---

## OS2CHAR Major Code: 0X0018 Minor Code: 267 (0X010B)

### Description

KbdPeek Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDPEEK

### Minor Code

267 (0X010B)

### Trace Groups

KBD

### Trace Types

PRE

### Traced Parameters

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

CharDataRec addr. = %A

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 268 (0X010C)

### Description

KbdRegister Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDREGISTER

### Minor Code

268 (0X010C)

### Trace Groups

KBD

### Trace Types

PRE

### Traced Parameters

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

Function Mask = %D

EntryName addr. = %A

EntryName = %S

ModuleName addr. = %A

ModuleName = %S

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 269 (0X010D)

### Description

KbdSetCP Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCP

### Minor Code

269 (0X010D)

### Trace Groups

KBD

### Trace Types

PRE

**Traced Parameters**

Major = %X   Minor = %Y  
ProcStatus = %B  
ProcType   = %B  
SessionID = %W  
KbdHandle = %W  
CodePageID = %W

-----

OS2CHAR Major Code: 0X0018 Minor Code: 270 (0X010E)

**Description**

KbdSetCustXT Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCUSTXT

**Minor Code**

270 (0X010E)

**Trace Groups**

KBD

**Trace Types**

PRE

**Traced Parameters**

Major = %X   Minor = %Y  
ProcStatus = %B  
ProcType   = %B  
SessionID = %W  
KbdHandle = %W  
XlateTable addr. = %A

-----

OS2CHAR Major Code: 0X0018 Minor Code: 271 (0X010F)

**Description**

KbdSetFgnd Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETFGND

**Minor Code**

271 (0X010F)

**Trace Groups**

KBD

**Trace Types**

PRE

**Traced Parameters**

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

OS2CHAR Major Code: 0X0018 Minor Code: 272 (0X0110)

**Description**

KbdSetStatus Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETSTATUS

**Minor Code**

272 (0X0110)

**Trace Groups**

KBD

**Trace Types**

PRE

**Traced Parameters**

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

Status addr. = %A

Status = %W

%W

%W

%W

%W

OS2CHAR Major Code: 0X0018 Minor Code: 273 (0X0111)

<u><b>Description</b></u>	KbdShellInit Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDSHELLINIT
<u><b>Minor Code</b></u>	273 (0X0111)
<u><b>Trace Groups</b></u>	KBD
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Major = %X   Minor = %Y ProcStatus = %B ProcType   = %B SessionID = %W

## OS2CHAR Major Code: 0X0018 Minor Code: 274 (0X0112)

<u><b>Description</b></u>	KbdStringIn Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDSTRINGIN
<u><b>Minor Code</b></u>	274 (0X0112)
<u><b>Trace Groups</b></u>	KBD
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	Major = %X   Minor = %Y ProcStatus = %B ProcType   = %B SessionID = %W KbdHandle = %W IOWAIT        = %W LengthBuffer addr. = %A CharBuffer addr.   = %A

CharBuffer length = %W

---

## OS2CHAR Major Code: 0X0018 Minor Code: 275 (0X0113)

### Description

KbdSynch Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSYNCH

### Minor Code

275 (0X0113)

### Trace Groups

KBD

### Trace Types

PRE

### Traced Parameters

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W

IOWAIT = %W

---

## OS2CHAR Major Code: 0X0018 Minor Code: 276 (0X0114)

### Description

KbdXlate Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDXLATE

### Minor Code

276 (0X0114)

### Trace Groups

KBD

### Trace Types

PRE

### Traced Parameters

Major = %X Minor = %Y

ProcStatus = %B

ProcType = %B

SessionID = %W  
KbdHandle = %W  
XlateRecord addr. = %A  
XlateRecord     = %B %B %B %B  
%B %B %B %B  
%B %B %B %B  
%B %B %B %B  
%B %B

---

## OS2CHAR Major Code: 0X0018 Minor Code: 513 (0X0201)

<u>Description</u>	(MOU) MouClose Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUCLOSE
<u>Minor Code</u>	513 (0X0201)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	ProcStatus = %b
	ProcType  = %b
	SessionID = %w
	MouHandle = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 514 (0X0202)

<u>Description</u>	(MOU) MouDeRegister Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUDEREGISTER
<u>Minor Code</u>	514 (0X0202)
<u>Trace Groups</u>	MOU



**Trace Types**  
PRE

**Traced Parameters**  
  
ProcStatus = %b  
ProcType = %b  
SessionID = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 515 (0X0203)

**Description**  
(MOU) MouDrawPtr Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.MOUDRAWPTR

**Minor Code**  
515 (0X0203)

**Trace Groups**  
MOU

**Trace Types**  
PRE

**Traced Parameters**  
  
ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 516 (0X0204)

**Description**  
(MOU) MouFlushQue Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.MOUFLUSHQUE

**Minor Code**  
516 (0X0204)

**Trace Groups**  
MOU

**Trace Types**  
PRE

**Traced Parameters**

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 517 (0X0205)

### Description

(MOU) MouGetDevStatus Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETDEVSTATUS

### Minor Code

517 (0X0205)

### Trace Groups

MOU

### Trace Types

PRE

### Traced Parameters

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
DevStatus addr. = %a

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 518 (0X0206)

### Description

(MOU) MouGetEventMask Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETEVENTMASK

### Minor Code

518 (0X0206)

### Trace Groups

MOU

### Trace Types

PRE

### Traced Parameters

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
EventMask addr. = %a

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 519 (0X0207)

**Description**

(MOU) MouGetNumButtons Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMBUTTONS

**Minor Code**

519 (0X0207)

**Trace Groups**

MOU

**Trace Types**

PRE

**Traced Parameters**

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
NumButtons addr. = %a

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 520 (0X0208)

**Description**

(MOU) MouGetNumMickeyes Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMMICKEYS

**Minor Code**

520 (0X0208)

**Trace Groups**

MOU

**Trace Types**

PRE

**Traced Parameters**

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
NumMickeyKeys addr. = %a

-----

OS2CHAR Major Code: 0X0018 Minor Code: 521 (0X0209)

**Description**

(MOU) MouGetNumQueEI Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMQUEEL

**Minor Code**

521 (0X0209)

**Trace Groups**

MOU

**Trace Types**

PRE

**Traced Parameters**

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
QueDataRec addr. = %a

-----

OS2CHAR Major Code: 0X0018 Minor Code: 522 (0X020A)

**Description**

(MOU) MouGetPtrPos Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRPOS

**Minor Code**

522 (0X020A)

**Trace Groups**

MOU

**Trace Types**

PRE

**Traced Parameters**

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

PtrPos addr. = %a

-----

OS2CHAR Major Code: 0X0018 Minor Code: 523 (0X020B)

**Description**

(MOU) MouGetPtrShape Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRSHAPE

**Minor Code**

523 (0X020B)

**Trace Groups**

MOU

**Trace Types**

PRE

**Traced Parameters**

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

PtrDefRec addr. = %a

PtrBuffer addr. = %a

PtrBuffer length = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 524 (0X020C)

**Description**

(MOU) MouGetScaleFact Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETSCALEFACT

**Minor Code** 524 (0X020C)

**Trace Groups** MOU

**Trace Types** PRE

**Traced Parameters**

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

ScaleStruc addr. = %a

-----

OS2CHAR Major Code: 0X0018 Minor Code: 525 (0X020D)

**Description** (MOU) MouGetThreshold Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETTHRESHOLD

**Minor Code** 525 (0X020D)

**Trace Groups** MOU

**Trace Types** PRE

**Traced Parameters**

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

ThresholdStruc addr. = %a

ThresholdStruc Length = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 526 (0X020E)

**Description** (MOU) MouInitReal Pre-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUINITREAL
<u>Minor Code</u>	526 (0X020E)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	ProcStatus = %b
	ProcType = %b
	SessionID = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 527 (0X020F)

<u>Description</u>	(MOU) MouOpen Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUOPEN
<u>Minor Code</u>	527 (0X020F)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	MouHandle addr. = %a
	PtrName addr. = %a
	PtrName = %s

## OS2CHAR Major Code: 0X0018 Minor Code: 528 (0X0210)

<u>Description</u>	(MOU) MouReadEventQue Pre-Invocation
--------------------	--------------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUREADEVENTQUE
<u>Minor Code</u>	528 (0X0210)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	MouHandle = %w
	EventBuf addr. = %a
	ReadType addr. = %a
	ReadType = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 529 (0X0211)

<u>Description</u>	(MOU) MouRegister Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUREGISTER
<u>Minor Code</u>	529 (0X0211)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	Function Mask = %d
	EntryName addr. = %a
	EntryName = %s
	ModuleName addr. = %a
	ModuleName = %s



-----

## OS2CHAR Major Code: 0X0018 Minor Code: 530 (0X0212)

<u>Description</u>	(MOU) MouRemovePtr Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUREMOVEPTR
<u>Minor Code</u>	530 (0X0212)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	MouHandle = %w
	PtrRec addr. = %a
	PtrRec.UpLEFTROW = %w
	.UpLEFTCOL = %w
	.LoRIGHTROW = %w
	.LoRIGHTCOL = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 531 (0X0213)

<u>Description</u>	(MOU) MouSetDevStatus Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERDEVSTATUS
<u>Minor Code</u>	531 (0X0213)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
DevStatus addr. = %a  
DevStatus = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 532 (0X0214)

### Description

(MOU) MouSetEventMask Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSEREVENTMASK

### Minor Code

532 (0X0214)

### Trace Groups

MOU

### Trace Types

PRE

### Traced Parameters

ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
EventMask addr. = %a  
EventMask = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 533 (0X0215)

### Description

(MOU) MouSetPtrPos Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERPTRPOS

### Minor Code

533 (0X0215)

### Trace Groups

MOU

**Trace Types**  
PRE

**Traced Parameters**  
  
ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
PtrPos addr. = %a  
PtrPos.ROW = %w  
.COL = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 534 (0X0216)

**Description**  
(MOU) MouSetPtrShape Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERPTRSHAPE

**Minor Code**  
534 (0X0216)

**Trace Groups**  
MOU

**Trace Types**  
PRE

**Traced Parameters**  
  
ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
PtrBuffer addr. = %a  
PtrDefRec addr. = %a  
PtrDefRec.BUFFERLEN = %w  
.COL = %w  
.ROW = %w  
.COLOFFSET = %w  
.ROWOFFSET = %w

# OS2CHAR Major Code: 0X0018 Minor Code: 535 (0X0217)

Description	(MOU) MouSetScaleFact Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERSCALEFACT
Minor Code	535 (0X0217)
Trace Groups	MOU
Trace Types	PRE
Traced Parameters	<div>ProcStatus = %b</div> <div>ProcType = %b</div> <div>SessionID = %w</div> <div>MouHandle = %w</div> <div>ScaleFact addr. = %a</div> <div>ScaleFact.ROWSCALE = %w</div> <div>.COLSCALE = %w</div>

# OS2CHAR Major Code: 0X0018 Minor Code: 536 (0X0218)

Description	(MOU) MouSetThreshold Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERTHRESHOLD
Minor Code	536 (0X0218)
Trace Groups	MOU
Trace Types	PRE
Traced Parameters	<div>ProcStatus = %b</div> <div>ProcType = %b</div> <div>SessionID = %w</div>

MouHandle = %w  
ThresholdStruc addr. = %a  
ThresholdStruc.Length = %w  
.Level1 = %w  
.Lev1Mult = %w  
.Level2 = %w  
.Lev2Mult = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 537 (0X0219)

### Description

(MOU) MouShellInit Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSHELLINIT

### Minor Code

537 (0X0219)

### Trace Groups

MOU

### Trace Types

PRE

### Traced Parameters

ProcStatus = %b

ProcType = %b

SessionID = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 538 (0X021A)

### Description

(MOU) MouSynch Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSYNCH

### Minor Code

538 (0X021A)

### Trace Groups

MOU

### Trace Types

PRE

### Traced Parameters

ProcStatus = %b

ProcType = %b

SessionID = %w

WaitFlags = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32769 (0X8001)

### Description

(OS) VioAssociate Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOASSOCIATE\_POSTDT

### Minor Code

32769 (0X8001)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Return Code = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32770 (0X8002)

### Description

(OS) VioCreateLogFont Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATELOGFONT\_POSTDT

### Minor Code

32770 (0X8002)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Return Code = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32771 (0X8003)

<b>Description</b>	(OS) VioCreatePS Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOCREATEPS_POSTDT
<b>Minor Code</b>	32771 (0X8003)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	
	PS Handle = %w
	Return Code = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32772 (0X8004)

<b>Description</b>	(OS) VioDeleteSetID Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIODELETESETID_POSTDT
<b>Minor Code</b>	32772 (0X8004)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	
	Return Code = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32773 (0X8005)

<b>Description</b>	(OS) VioDeRegister Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIODEREGISTER_POSTDT
<b>Minor Code</b>	32773 (0X8005)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32774 (0X8006)

**Description**

(OS) VioDestroyPS Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIODESTROYPS\_POSTDT

**Minor Code**

32774 (0X8006)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32775 (0X8007)

**Description**

(OS) VioEndPopUp Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOENDPOPUP\_POSTDT

**Minor Code**

32775 (0X8007)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w



-----

# OS2CHAR Major Code: 0X0018 Minor Code: 32776 (0X8008)

Description	(OS) VioGetAnsi Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETANSI_POSTDT
Minor Code	32776 (0X8008)
Trace Groups	VIO
Trace Types	POST
Traced Parameters	
Handle	= %w
Indicator	= %w
Return Code	= %w

-----

# OS2CHAR Major Code: 0X0018 Minor Code: 32777 (0X8009)

Description	(OS) VioGetBuf Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETBUF_POSTDT
Minor Code	32777 (0X8009)
Trace Groups	VIO
Trace Types	POST
Traced Parameters	
Handle	= %w
Length	= %w
LVBPtr	= %d
Return Code	= %w

-----

# OS2CHAR Major Code: 0X0018 Minor Code: 32778 (0X800A)

<b><u>Description</u></b>	(OS) VioGetConfig Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCONFIG_POSTDT
<b><u>Minor Code</u></b>	32778 (0X800A)
<b><u>Trace Groups</u></b>	VIO
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
Handle	= %w
Config Data	= %r%w
Reserved	= %w
Return Code	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32779 (0X800B)

<b><u>Description</u></b>	(OS) VioGetCp Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCP_POSTDT
<b><u>Minor Code</u></b>	32779 (0X800B)
<b><u>Trace Groups</u></b>	VIO
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
Handle	= %w
Code Page Id	= %w
Reserved	= %w
Return Code	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32780 (0X800C)

<b>Description</b>	(OS) VioGetCurPos Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURPOS_POSTDT
<b>Minor Code</b>	32780 (0X800C)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	
	Handle = %w
	Column = %w
	Row = %w
	Return Code = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32781 (0X800D)

<b>Description</b>	(OS) VioGetCurType Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETCURTYPE_POSTDT
<b>Minor Code</b>	32781 (0X800D)
<b>Trace Groups</b>	VIO
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	
	Handle = %w
	Cursor Data = %w%w%w%w
	Return Code = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32782 (0X800E)

<u>Description</u>	(OS) VioGetDeviceCellSize Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETDEVICECELLSIZE_POSTDT
<u>Minor Code</u>	32782 (0X800E)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	PS Handle = %w
	Width = %w
	Height = %w
	Return Code = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32783 (0X800F)

<u>Description</u>	(OS) VioGetFont Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETFONT_POSTDT
<u>Minor Code</u>	32783 (0X800F)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Handle = %w
	Request Block = %r%w
	Return Code = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32784 (0X8010)

<u>Description</u>	(OS) VioGetMode Post-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETMODE\_POSTDT

**Minor Code**

32784 (0X8010)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle	= %w
ModeData	= %r%w
Return Code	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32785 (0X8011)

**Description**

(OS) VioGetOrg Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETORG\_POSTDT

**Minor Code**

32785 (0X8011)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

PS Handle	= %w
Column	= %w
Row	= %w
Return Code	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32786 (0X8012)

**Description**

(OS) VioGetPhysBuf Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPHYSBUF\_POSTDT

**Minor Code**

32786 (0X8012)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

DisplayBuf = %d%d%w%w  
Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32787 (0X8013)

**Description**

(OS) VioGetPSAddress Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETPSADDRESS\_POSTDT

**Minor Code**

32787 (0X8013)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

OS2CHAR Major Code: 0X0018 Minor Code: 32788 (0X8014)

**Description**

(OS) VioGetState Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGETSTATE\_POSTDT

**Minor Code**

32788 (0X8014)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Request Block       = %r%w  
Return Code         = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32789 (0X8015)

### Description

(OS) VioGlobalReg Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOGLOBALREG\_POSTDT

### Minor Code

32789 (0X8015)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Return Code       = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32790 (0X8016)

### Description

(OS) VioModeUndo Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEUNDO\_POSTDT

### Minor Code

32790 (0X8016)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Return Code       = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32791 (0X8017)

### Description

(OS) VioModeWait Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOMODEWAIT\_POSTDT

**Minor Code**

32791 (0X8017)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Notify Type           = %w

Return Code           = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32792 (0X8018)

**Description**

(OS) VioPopUp Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOPOPUP\_POSTDT

**Minor Code**

32792 (0X8018)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle               = %w

Return Code          = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32793 (0X8019)

**Description**

(OS) VioPrtSc Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSC\_POSTDT

**Minor Code**

32793 (0X8019)

**Trace Groups**

VIO



**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32794 (0X801A)

**Description**

(OS) VioPrtScToggle Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOPRTSCTOGGLE\_POSTDT

**Minor Code**

32794 (0X801A)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32795 (0X801B)

**Description**

(OS) VioQueryConsole Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYCONSOLE\_POSTDT

**Minor Code**

32795 (0X801B)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

# OS2CHAR Major Code: 0X0018 Minor Code: 32796 (0X801C)

<u>Description</u>	(OS) VioQueryFonts Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYFONTS_POSTDT
<u>Minor Code</u>	32796 (0X801C)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
PS Handle	= %w
Options	= %d
Fonts Returned	= %d
Metrics Length	= %d
Metrics - szFamilyname[FACESIZE]	= %s
szFacename[FACESIZE]	= %s
idRegistry	= %w
usCodePage	= %w
IEmHeight	= %d
IXHeight	= %d
IMaxAscender	= %d
IMaxDescender	= %d
ILowerCaseAscent	= %d
ILowerCaseDescent	= %d
InternalLeading	= %d
IEternalLeading	= %d
IAveCharWidth	= %d
IMaxCharInc	= %d
IEmInc	= %d
IMaxBaselineExt	= %d
sCharSlope	= %w
sInlineDir	= %w

sCharRot	= %w
usWeightClass	= %w
usWidthClass	= %w
sXDeviceRes	= %w
sYDeviceRes	= %w
sFirstChar	= %w
sLastChar	= %w
sDefaultChar	= %w
sBreakChar	= %w
sNominalPointSize	= %w
sMinimumPointSize	= %w
sMaximumPointSize	= %w
fsType	= %w
fsDefn	= %w
fsSelection	= %w
fsCapabilities	= %w
ISubscriptXSize	= %d
ISubscriptYSize	= %d
ISubscriptXOffset	= %d
ISubscriptYOffset	= %d
ISuperscriptXSize	= %d
ISuperscriptYSize	= %d
ISuperscriptXOffset	= %d
ISuperscriptYOffset	= %d
IUnderscoreSize	= %d
IUnderscorePosition	= %d
IStrikeoutSize	= %d
IStrikeoutPosition	= %d
sKerningPairs	= %w
sFamilyClass	= %w
IMatch	= %d
Fonts Not Returned	= %d
Return Code	= %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32797

# (0X801D)

Description	(OS) VioQuerySetIDs Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOQUERYSETIDS_POSTDT
Minor Code	32797 (0X801D)
Trace Groups	VIO
Trace Types	POST
Traced Parameters	
PS Handle	= %w
Count	= %d
Types	= %d
Font Names	= %r%b
Local Identifiers	= %d
Return Code	= %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32798 (0X801E)

Description	(OS) VioReadCellStr Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCELLSTR_POSTDT
Minor Code	32798 (0X801E)
Trace Groups	VIO
Trace Types	POST
Traced Parameters	
Handle	= %w
Column	= %w
Row	= %w
Length	= %w
CellStr	= %r%b

Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32799 (0X801F)

### Description

(OS) VioReadCharStr Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOREADCHARSTR\_POSTDT

### Minor Code

32799 (0X801F)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Handle = %w  
Column = %w  
Row = %w  
Length = %w  
CharStr = %r%b  
Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32800 (0X8020)

### Description

(OS) VioRegister Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOREGISTER\_POSTDT

### Minor Code

32800 (0X8020)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Return Code = %w

---

# OS2CHAR Major Code: 0X0018 Minor Code: 32801 (0X8021)

Description	(OS) VioSavRedrawUndo Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWUNDO_POSTDT
Minor Code	32801 (0X8021)
Trace Groups	VIO
Trace Types	POST
Traced Parameters	
	Return Code = %w

# OS2CHAR Major Code: 0X0018 Minor Code: 32802 (0X8022)

Description	(OS) VioSavRedrawWait Post-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSAVREDRAWWAIT_POSTDT
Minor Code	32802 (0X8022)
Trace Groups	VIO
Trace Types	POST
Traced Parameters	
	Reserved = %w
	Notify Type = %w
	SavRedrawIndic = %w
	Return Code = %w

# OS2CHAR Major Code: 0X0018 Minor Code: 32803 (0X8023)

Description

(OS) VioScrLock Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCRLOCK\_POSTDT

**Minor Code**

32803 (0X8023)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Status = %b

Wait Flag = %w

Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32804 (0X8024)

**Description**

(OS) VioScrollDn Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLDN\_POSTDT

**Minor Code**

32804 (0X8024)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32805 (0X8025)

**Description**

(OS) VioScrollLf Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLLF\_POSTDT

**Minor Code**

32805 (0X8025)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32806 (0X8026)

**Description**

(OS) VioScrollRt Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLRT\_POSTDT

**Minor Code**

32806 (0X8026)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32807 (0X8027)

**Description**

(OS) VioScrollUp Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCROLLUP\_POSTDT

**Minor Code**

32807 (0X8027)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**



Handle = %w  
Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32808 (0X8028)

### Description

(OS) VioScrUnLock Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSCRUNLOCK\_POSTDT

### Minor Code

32808 (0X8028)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Handle = %w  
Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32809 (0X8029)

### Description

(OS) VioSetAnsi Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETANSI\_POSTDT

### Minor Code

32809 (0X8029)

### Trace Groups

VIO

### Trace Types

POST

### Traced Parameters

Handle = %w  
Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32810 (0X802A)

<u>Description</u>	(OS) VioSetCp Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCP_POSTDT
<u>Minor Code</u>	32810 (0X802A)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
Handle	= %w
Return Code	= %w

OS2CHAR Major Code: 0X0018 Minor Code: 32811 (0X802B)

<u>Description</u>	(OS) VioSetCurPos Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURPOS_POSTDT
<u>Minor Code</u>	32811 (0X802B)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
Handle	= %w
Return Code	= %w

OS2CHAR Major Code: 0X0018 Minor Code: 32812 (0X802C)

<u>Description</u>	(OS) VioSetCurType Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETCURTYPE_POSTDT

<u>Minor Code</u>	32812 (0X802C)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
Handle	= %w
Return Code	= %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32813 (0X802D)

<u>Description</u>	(OS) VioSetDeviceCellSize Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETDEVICECELLSIZE_POSTDT
<u>Minor Code</u>	32813 (0X802D)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
Return Code	= %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 32814 (0X802E)

<u>Description</u>	(OS) VioSetFont Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETFONT_POSTDT
<u>Minor Code</u>	32814 (0X802E)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST

**Traced Parameters**

Handle = %w  
Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32815 (0X802F)

**Description**

(OS) VioSetMode Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETMODE\_POSTDT

**Minor Code**

32815 (0X802F)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w  
Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32816 (0X8030)

**Description**

(OS) VioSetOrg Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETORG\_POSTDT

**Minor Code**

32816 (0X8030)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32817 (0X8031)

<u>Description</u>	(OS) VioSetState Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSETSTATE_POSTDT
<u>Minor Code</u>	32817 (0X8031)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Handle = %w
	Return Code = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32818 (0X8032)

<u>Description</u>	(OS) VioShieldInit Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDINIT_POSTDT
<u>Minor Code</u>	32818 (0X8032)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 32819 (0X8033)

<u>Description</u>	(OS) VioShieldTerm Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHIELDTERM_POSTDT
<u>Minor Code</u>	32819 (0X8033)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32820 (0X8034)

**Description**

(OS) VioShowBuf Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWBUF\_POSTDT

**Minor Code**

32820 (0X8034)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32821 (0X8035)

**Description**

(OS) VioShowPS Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOSHOWPS\_POSTDT

**Minor Code**

32821 (0X8035)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32822 (0X8036)

<u>Description</u>	(OS) VioWrtCellStr Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCELLSTR_POSTDT
<u>Minor Code</u>	32822 (0X8036)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
Handle	= %w
Return Code	= %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32823 (0X8037)

<u>Description</u>	(OS) VioWrtCharStr Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCHARSTR_POSTDT
<u>Minor Code</u>	32823 (0X8037)
<u>Trace Groups</u>	VIO
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
Handle	= %w
Return Code	= %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 32824 (0X8038)

<u>Description</u>	(OS) VioWrtCharStrAtt Post-Invocation
--------------------	---------------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTCHARSTRATT_POSTDT		
<u>Minor Code</u>	32824 (0X8038)		
<u>Trace Groups</u>	VIO		
<u>Trace Types</u>	POST		
<u>Traced Parameters</u>			
	Handle	=	%w
	Return Code	=	%w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32825 (0X8039)

<u>Description</u>	(OS) VioWrtNAttr Post-Invocation		
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNATTR_POSTDT		
<u>Minor Code</u>	32825 (0X8039)		
<u>Trace Groups</u>	VIO		
<u>Trace Types</u>	POST		
<u>Traced Parameters</u>			
	Handle	=	%w
	Return Code	=	%w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32826 (0X803A)

<u>Description</u>	(OS) VioWrtNCell Post-Invocation		
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNCELL_POSTDT		
<u>Minor Code</u>	32826 (0X803A)		
<u>Trace Groups</u>	VIO		



**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32827 (0X803B)

**Description**

(OS) VioWrtNChar Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTNCHAR\_POSTDT

**Minor Code**

32827 (0X803B)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 32828 (0X803C)

**Description**

(OS) VioWrtTTY Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.VIOWRTTTY\_POSTDT

**Minor Code**

32828 (0X803C)

**Trace Groups**

VIO

**Trace Types**

POST

**Traced Parameters**

Handle = %w

Return Code = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33025 (0X8101)

### Description

KbdCharIn Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDCHARIN

### Minor Code

33025 (0X8101)

### Trace Groups

KBD

### Trace Types

POST

### Traced Parameters

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

CharDataRec.charcode = %B

.scancode = %B

.status = %B

.NLSstat = %B

.shiftstat = %W

.timestamp = %D

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33026 (0X8102)

### Description

KbdClose Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDCLOSE

### Minor Code

33026 (0X8102)

### Trace Groups

	KBD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Major = %X  Minor = %Y
	Return Code = %W
	ProcStatus = %B
	ProcType  = %B
	SessionID = %W
	KbdHandle = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33027 (0X8103)

<u>Description</u>	KbdGetHWID Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETHWID
<u>Minor Code</u>	33027 (0X8103)
<u>Trace Groups</u>	KBD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Major = %X  Minor = %Y
	Return Code = %W
	ProcStatus = %B
	ProcType  = %B
	SessionID = %W
	KbdHandle = %W
	HWIDStruc = %W
	= %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33028 (0X8104)

<u>Description</u>	KbdDeRegister Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDDEREGISTER
<u>Minor Code</u>	33028 (0X8104)
<u>Trace Groups</u>	KBD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	<p>Major = %X   Minor = %Y</p> <p>Return Code = %W</p> <p>ProcStatus = %B</p> <p>ProcType   = %B</p> <p>SessionID = %W</p>

## OS2CHAR Major Code: 0X0018 Minor Code: 33029 (0X8105)

<u>Description</u>	KbdFlushBuffer Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDFLUSHBUFFER
<u>Minor Code</u>	33029 (0X8105)
<u>Trace Groups</u>	KBD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	<p>Major = %X   Minor = %Y</p> <p>Return Code = %W</p> <p>ProcStatus = %B</p> <p>ProcType   = %B</p> <p>SessionID = %W</p> <p>KbdHandle = %W</p>

## OS2CHAR Major Code: 0X0018 Minor Code: 33030 (0X8106)

<b><u>Description</u></b>	KbdFreeFocus Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDFREEFOCUS
<b><u>Minor Code</u></b>	33030 (0X8106)
<b><u>Trace Groups</u></b>	KBD
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	<p>Major = %X   Minor = %Y</p> <p>Return Code = %W</p> <p>ProcStatus = %B</p> <p>ProcType   = %B</p> <p>SessionID = %W</p> <p>KbdHandle = %W</p>

## OS2CHAR Major Code: 0X0018 Minor Code: 33031 (0X8107)

<b><u>Description</u></b>	KbdGetCP Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETCP
<b><u>Minor Code</u></b>	33031 (0X8107)
<b><u>Trace Groups</u></b>	KBD
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	<p>Major = %X   Minor = %Y</p> <p>Return Code = %W</p> <p>ProcStatus = %B</p> <p>ProcType   = %B</p> <p>SessionID = %W</p> <p>KbdHandle = %W</p>

CodePageID = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33032 (0X8108)

### Description

KbdGetFocus Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETFOCUS

### Minor Code

33032 (0X8108)

### Trace Groups

KBD

### Trace Types

POST

### Traced Parameters

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33033 (0X8109)

### Description

KbdGetStatus Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDGETSTATUS

### Minor Code

33033 (0X8109)

### Trace Groups

KBD

### Trace Types

POST

### Traced Parameters

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

Status = %W

%W

%W

%W

%W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33034 (0X810A)

### Description

KbdOpen Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDOPEN

### Minor Code

33034 (0X810A)

### Trace Groups

KBD

### Trace Types

POST

### Traced Parameters

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33035 (0X810B)

### Description

KbdPeek Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDPEEK

**Minor Code**

33035 (0X810B)

**Trace Groups**

KBD

**Trace Types**

POST

**Traced Parameters**

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

CharDataRec.charcode = %B

.scancode = %B

.status = %B

.NLSstat = %B

.shiftstat = %W

.timestamp = %D

---

## OS2CHAR Major Code: 0X0018 Minor Code: 33036 (0X810C)

**Description**

KbdRegister Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDREGISTER

**Minor Code**

33036 (0X810C)

**Trace Groups**

KBD

**Trace Types**

POST

**Traced Parameters**

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B



SessionID = %W

---

## OS2CHAR Major Code: 0X0018 Minor Code: 33037 (0X810D)

### Description

KbdSetCP Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCP

### Minor Code

33037 (0X810D)

### Trace Groups

KBD

### Trace Types

POST

### Traced Parameters

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

---

## OS2CHAR Major Code: 0X0018 Minor Code: 33038 (0X810E)

### Description

KbdSetCustXT Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETCUSTXT

### Minor Code

33038 (0X810E)

### Trace Groups

KBD

### Trace Types

POST

### Traced Parameters

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33039 (0X810F)

### Description

KbdSetFgnd Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETFGND

### Minor Code

33039 (0X810F)

### Trace Groups

KBD

### Trace Types

POST

### Traced Parameters

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33040 (0X8110)

### Description

KbdSetStatus Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSETSTATUS

### Minor Code

33040 (0X8110)

### Trace Groups

KBD

### Trace Types

POST

**Traced Parameters**

Major = %X   Minor = %Y  
Return Code = %W  
ProcStatus = %B  
ProcType  = %B  
SessionID = %W  
KbdHandle = %W

-----

OS2CHAR Major Code: 0X0018 Minor Code: 33041 (0X8111)

**Description**

KbdShellInit Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSHELLINIT

**Minor Code**

33041 (0X8111)

**Trace Groups**

KBD

**Trace Types**

POST

**Traced Parameters**

Major = %X   Minor = %Y  
Return Code = %W  
ProcStatus = %B  
ProcType  = %B  
SessionID = %W

-----

OS2CHAR Major Code: 0X0018 Minor Code: 33042 (0X8112)

**Description**

KbdStringIn Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.KBDSTRINGIN

**Minor Code**

33042 (0X8112)

**Trace Groups**

KBD

**Trace Types** POST

**Traced Parameters**

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

KbdHandle = %W

CharBuffer length = %W

CharBuffer = %S

**Note:** CharBuffer was added with OS/2 Warp V3.0 fix pack 41 and OS/2 Warp V4.0 fix pack 10.

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33043 (0X8113)

**Description** KbdSynch Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2CHAR.KBDSYNCH

**Minor Code** 33043 (0X8113)

**Trace Groups** KBD

**Trace Types** POST

**Traced Parameters**

Major = %X Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType = %B

SessionID = %W

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33044 (0X8114)

<b>Description</b>	KbdXlate Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.KBDXLATE
<b>Minor Code</b>	33044 (0X8114)
<b>Trace Groups</b>	KBD
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	

Major = %X    Minor = %Y

Return Code = %W

ProcStatus = %B

ProcType    = %B

SessionID = %W

KbdHandle = %W

XlateRecord = %B %B %B %B

%B %B %B %B

%B %B %B %B

%B %B %B %B

%B %B

## OS2CHAR Major Code: 0X0018 Minor Code: 33281 (0X8201)

<b>Description</b>	(MOU) MouClose Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUCLOSE
<b>Minor Code</b>	33281 (0X8201)
<b>Trace Groups</b>	MOU
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	
	Return Code = %w
	ProcStatus = %b
	ProcType    = %b

SessionID = %w

MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33282 (0X8202)

### Description

(MOU) MouDeRegister Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUDEREGISTER

### Minor Code

33282 (0X8202)

### Trace Groups

MOU

### Trace Types

POST

### Traced Parameters

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33283 (0X8203)

### Description

(MOU) MouDrawPtr Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUDRAWPTR

### Minor Code

33283 (0X8203)

### Trace Groups

MOU

### Trace Types

POST

### Traced Parameters

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33284 (0X8204)

### Description

(MOU) MouFlushQue Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUFLUSHQUE

### Minor Code

33284 (0X8204)

### Trace Groups

MOU

### Trace Types

POST

### Traced Parameters

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33285 (0X8205)

### Description

(MOU) MouGetDevStatus Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETDEVSTATUS

### Minor Code

33285 (0X8205)

### Trace Groups

MOU

### Trace Types

POST

### Traced Parameters

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

DevStatus = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33286 (0X8206)

### Description

(MOU) MouGetEventMask Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETEVENTMASK

### Minor Code

33286 (0X8206)

### Trace Groups

MOU

### Trace Types

POST

### Traced Parameters

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

EventMask = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33287 (0X8207)

### Description

(MOU) MouGetNumButtons Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMBUTTONS

### Minor Code

33287 (0X8207)

### Trace Groups

MOU

### Trace Types

POST



**Traced Parameters**

Return Code = %w  
ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
NumButtons = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 33288 (0X8208)

**Description**

(MOU) MouGetNumMickeyKeys Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMMICKEYS

**Minor Code**

33288 (0X8208)

**Trace Groups**

MOU

**Trace Types**

POST

**Traced Parameters**

Return Code = %w  
ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
NumMickeyKeys = %w

-----

OS2CHAR Major Code: 0X0018 Minor Code: 33289 (0X8209)

**Description**

(MOU) MouGetNumQueueEl Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETNUMQUEEL

**Minor Code**

33289 (0X8209)

**Trace Groups**

MOU

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

Que Events = %w

Que Max Events = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33290 (0X820A)

**Description**

(MOU) MouGetPtrPos Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRPOS

**Minor Code**

33290 (0X820A)

**Trace Groups**

MOU

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

PtrPos.ROW = %w

.COL = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33291 (0X820B)

<b><u>Description</u></b>	(MOU) MouGetPtrShape Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETPTRSHAPE
<b><u>Minor Code</u></b>	33291 (0X820B)
<b><u>Trace Groups</u></b>	MOU
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	<div>Return Code = %w</div> <div>ProcStatus = %b</div> <div>ProcType = %b</div> <div>SessionID = %w</div> <div>MouHandle = %w</div> <div>PtrDefRec.LENGTH = %w</div> <div>.COL = %w</div> <div>.ROW = %w</div> <div>.COLOFFSET = %w</div> <div>.ROWOFFSET = %w</div>

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33292 (0X820C)

<b><u>Description</u></b>	(MOU) MouGetScaleFact Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETSCALEFACT
<b><u>Minor Code</u></b>	33292 (0X820C)
<b><u>Trace Groups</u></b>	MOU
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	<div>Return Code = %w</div> <div>ProcStatus = %b</div> <div>ProcType = %b</div>

SessionID = %w  
MouHandle = %w  
ScaleFact.ROWSCALE = %w  
.COLSCALE = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 33293 (0X820D)

### Description

(MOU) MouGetThreshold Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUGETTHRESHOLD

### Minor Code

33293 (0X820D)

### Trace Groups

MOU

### Trace Types

POST

### Traced Parameters

Return Code = %w  
ProcStatus = %b  
ProcType = %b  
SessionID = %w  
MouHandle = %w  
ThresholdStruc.Length = %w  
.Level1 = %w  
.Lev1Mult = %w  
.Level2 = %w  
.Lev2Mult = %w

---

## OS2CHAR Major Code: 0X0018 Minor Code: 33294 (0X820E)

### Description

(MOU) MouInitReal Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: OS2CHAR.MOUINTREAL

**Minor Code** 33294 (0X820E)

**Trace Groups** MOU

**Trace Types** POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33295 (0X820F)

**Description** (MOU) MouOpen Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2CHAR.MOUOPEN

**Minor Code** 33295 (0X820F)

**Trace Groups** MOU

**Trace Types** POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33296 (0X8210)

**Description** (MOU) MouReadEventQue Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2CHAR.MOUREADEVENTQUE

**Minor Code** 33296 (0X8210)

**Trace Groups** MOU

**Trace Types** POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

EventBuffer.MOUSTATE = %w

.EVENTTIME = %d

.ROW = %w

.COL = %w

ReadType = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33297 (0X8211)

**Description** (MOU) MouRegister Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: OS2CHAR.MOUREGISTER

**Minor Code** 33297 (0X8211)

**Trace Groups** MOU

**Trace Types** POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33298 (0X8212)

<u>Description</u>	(MOU) MouRemovePtr Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUREMOVEPTR
<u>Minor Code</u>	33298 (0X8212)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33299 (0X8213)

<u>Description</u>	(MOU) MouSetDevStatus Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERDEVSTATUS
<u>Minor Code</u>	33299 (0X8213)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33300 (0X8214)

<b>Description</b>	(MOU) MouSetEventMask Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSEREVENTMASK
<b>Minor Code</b>	33300 (0X8214)
<b>Trace Groups</b>	MOU
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Return Code = %w ProcStatus = %b ProcType = %b SessionID = %w MouHandle = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 33301 (0X8215)

<b>Description</b>	(MOU) MouSetPtrPos Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERPTRPOS
<b>Minor Code</b>	33301 (0X8215)
<b>Trace Groups</b>	MOU
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	Return Code = %w ProcStatus = %b ProcType = %b SessionID = %w MouHandle = %w

## OS2CHAR Major Code: 0X0018 Minor Code: 33302 (0X8216)



<u>Description</u>	(MOU) MouSetPtrShape Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERPTRSHAPE
<u>Minor Code</u>	33302 (0X8216)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33303 (0X8217)

<u>Description</u>	(MOU) MouSetScaleFact Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERSCALEFACT
<u>Minor Code</u>	33303 (0X8217)
<u>Trace Groups</u>	MOU
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w
	ProcStatus = %b
	ProcType = %b
	SessionID = %w
	MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33304 (0X8218)

**Description**

(MOU) MouSetThreshold Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSERTHRESHOLD

**Minor Code**

33304 (0X8218)

**Trace Groups**

MOU

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

MouHandle = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33305 (0X8219)

**Description**

(MOU) MouShellInit Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: OS2CHAR.MOUSHELLINIT

**Minor Code**

33305 (0X8219)

**Trace Groups**

MOU

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

ProcStatus = %b

ProcType = %b

SessionID = %w

-----

## OS2CHAR Major Code: 0X0018 Minor Code: 33306 (0X821A)

<b>Description</b>	(MOU) MouSynch Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: OS2CHAR.MOUSYNCH
<b>Minor Code</b>	33306 (0X821A)
<b>Trace Groups</b>	MOU
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	
	Return Code = %w
	ProcStatus = %b
	ProcType = %b
	SessionID = %w

## QUECALLS.DLL Trace Events

The tracepoints for the QUECALLS.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for QUECALLS Major Code: 0X0016, sorted by minor code.](#)  
[Trace events for QUECALLS Major Code: 0X0016 ,sorted by tracepoint.](#)  
[QUECALLS API Tracepoints Indirected Via DOSCALL1.](#)

## Trace Events for QUECALLS Major Code: 0X0016, Sorted by Minor Code

00001 (0X0001) (OS) DosCloseQueue Pre-Invocation  
00002 (0X0002) (OS) DosCreateQueue Pre-Invocation  
00003 (0X0003) (OS) DosOpenQueue Pre-Invocation  
00004 (0X0004) (OS) DosPeekQueue Pre-Invocation  
00005 (0X0005) (OS) DosPurgeQueue Pre-Invocation  
00006 (0X0006) (OS) DosQueryQueue Pre-Invocation  
00007 (0X0007) (OS) DosReadQueue Pre-Invocation  
00008 (0X0008) (OS) DosWriteQueue Pre-Invocation  
00009 (0X0009) (OS) Peek Data Packet From Queue  
00010 (0X000A) (OS) Read Data Packet From Queue  
00011 (0X000B) (OS) Write Data Packet To Queue  
32769 (0X8001) (OS) DosCloseQueue Post-Invocation  
32770 (0X8002) (OS) DosCreateQueue Post-Invocation

32771 (0X8003) (OS) DosOpenQueue Post-Invocation  
32772 (0X8004) (OS) DosPeekQueue Post-Invocation  
32773 (0X8005) (OS) DosPurgeQueue Post-Invocation  
32774 (0X8006) (OS) DosQueryQueue Post-Invocation  
32775 (0X8007) (OS) DosReadQueue Post-Invocation  
32776 (0X8008) (OS) DosWriteQueue Post-Invocation

---

## Trace Events for QUECALLS Major Code: 0X0016, Sorted by Tracepoint

(OS) DosCloseQueue Post-Invocation 32769 (0X8001)  
(OS) DosCloseQueue Pre-Invocation 00001 (0X0001)  
(OS) DosCreateQueue Post-Invocation 32770 (0X8002)  
(OS) DosCreateQueue Pre-Invocation 00002 (0X0002)  
(OS) DosOpenQueue Post-Invocation 32771 (0X8003)  
(OS) DosOpenQueue Pre-Invocation 00003 (0X0003)  
(OS) DosPeekQueue Post-Invocation 32772 (0X8004)  
(OS) DosPeekQueue Pre-Invocation 00004 (0X0004)  
(OS) DosPurgeQueue Post-Invocation 32773 (0X8005)  
(OS) DosPurgeQueue Pre-Invocation 00005 (0X0005)  
(OS) DosQueryQueue Post-Invocation 32774 (0X8006)  
(OS) DosQueryQueue Pre-Invocation 00006 (0X0006)  
(OS) DosReadQueue Post-Invocation 32775 (0X8007)  
(OS) DosReadQueue Pre-Invocation 00007 (0X0007)  
(OS) DosWriteQueue Post-Invocation 32776 (0X8008)  
(OS) DosWriteQueue Pre-Invocation 00008 (0X0008)  
(OS) Peek Data Packet From Queue 00009 (0X0009)  
(OS) Read Data Packet From Queue 00010 (0X000A)  
(OS) Write Data Packet To Queue 00011 (0X000B)

---

## QUECALLS Major Code: 0X0016 Minor Code: 1 (0X0001)

### Description

(OS) DosCloseQueue Pre-Invocation

### Tracepoint

Source line defined dynamic tracepoint: @closeq.c in QUECALLS.

### Minor Code

1 (0X0001)

### Trace Groups

No groups assigned.

### Trace Types

PRE

### Traced Parameters

Handle=%w

---

## QUECALLS Major Code: 0X0016 Minor Code: 2 (0X0002)

<b><u>Description</u></b>	(OS) DosCreateQueue Pre-Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @createq.c in QUECALLS.
<b><u>Minor Code</u></b>	2 (0X0002)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Priority=%w Queue name=%s

-----

## QUECALLS Major Code: 0X0016 Minor Code: 3 (0X0003)

<b><u>Description</u></b>	(OS) DosOpenQueue Pre-Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @openq.c in QUECALLS.
<b><u>Minor Code</u></b>	3 (0X0003)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Queue name=%s

-----

## QUECALLS Major Code: 0X0016 Minor Code: 4 (0X0004)

<b><u>Description</u></b>	(OS) DosPeekQueue Pre-Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @peekq.c in QUECALLS.
<b><u>Minor Code</u></b>	4 (0X0004)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Handle=%w Element Code=%w

Semaphore Handle=%d No Wait Flag=%b%i1

-----

QUECALLS Major Code: 0X0016 Minor Code: 5 (0X0005)

**Description** (OS) DosPurgeQueue Pre-Invocation

**Tracepoint** Source line defined dynamic tracepoint: @purgeq.c in QUECALLS.

**Minor Code** 5 (0X0005)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Handle=%w

-----

QUECALLS Major Code: 0X0016 Minor Code: 6 (0X0006)

**Description** (OS) DosQueryQueue Pre-Invocation

**Tracepoint** Source line defined dynamic tracepoint: @queryq.c in QUECALLS.

**Minor Code** 6 (0X0006)

**Trace Groups** No groups assigned.

**Trace Types** PRE

**Traced Parameters**

Handle=%w

-----

# QUECALLS Major Code: 0X0016 Minor Code: 7 (0X0007)

Description	(OS) DosReadQueue Pre-Invocation
Tracepoint	Source line defined dynamic tracepoint: @readq.c in QUECALLS.
Minor Code	7 (0X0007)
Trace Groups	No groups assigned.
Trace Types	PRE
Traced Parameters	Handle=%w Element Code=%w Semaphore Handle=%d No Wait Flag=%b

# QUECALLS Major Code: 0X0016 Minor Code: 8 (0X0008)

Description	(OS) DosWriteQueue Pre-Invocation
Tracepoint	Source line defined dynamic tracepoint: @writeq.c in QUECALLS.
Minor Code	8 (0X0008)
Trace Groups	No groups assigned.
Trace Types	PRE
Traced Parameters	Handle=%w Request=%d Length=%d Address=%a Priority=%b

# QUECALLS Major Code: 0X0016 Minor Code: 9 (0X0009)

Description	(OS) Peek Data Packet From Queue
Tracepoint	

Source line defined dynamic tracepoint: @peekq.c in QUECALLS.

**Minor Code**

9 (0X0009)

**Trace Groups**

No groups assigned.

**Trace Types**

INT

**Traced Parameters**

Length=%w Address=%a Element Code=%w

-----

## QUECALLS Major Code: 0X0016 Minor Code: 10 (0X000A)

**Description**

(OS) Read Data Packet From Queue

**Tracepoint**

Source line defined dynamic tracepoint: @readq.c in QUECALLS.

**Minor Code**

10 (0X000A)

**Trace Groups**

No groups assigned.

**Trace Types**

INT

**Traced Parameters**

Length=%d Address=%a Element Code=%w

-----

## QUECALLS Major Code: 0X0016 Minor Code: 11 (0X000B)

**Description**

(OS) Write Data Packet To Queue

**Tracepoint**

Source line defined dynamic tracepoint: @writeq.c in QUECALLS.

**Minor Code**

11 (0X000B)

**Trace Groups**

No groups assigned.

**Trace Types**

INT

**Traced Parameters**



Length=%d Address=%a Element Code=%w

---

## QUECALLS Major Code: 0X0016 Minor Code: 32769 (0X8001)

**Description**

(OS) DosCloseQueue Post-Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @closeq.c in QUECALLS.

**Minor Code**

32769 (0X8001)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code=%w

---

## QUECALLS Major Code: 0X0016 Minor Code: 32770 (0X8002)

**Description**

(OS) DosCreateQueue Post-Invocation

**Tracepoint**

Source line defined dynamic tracepoint: @createq.c in QUECALLS.

**Minor Code**

32770 (0X8002)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code=%w Handle=%w

---

## QUECALLS Major Code: 0X0016 Minor Code: 32771

## (0X8003)

<u>Description</u>	(OS) DosOpenQueue Post-Invocation
<u>Tracepoint</u>	Source line defined dynamic tracepoint: @openq.c in QUECALLS.
<u>Minor Code</u>	32771 (0X8003)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code=%w Handle=%w Owner PID=%w

-----

## QUECALLS Major Code: 0X0016 Minor Code: 32772 (0X8004)

<u>Description</u>	(OS) DosPeekQueue Post-Invocation
<u>Tracepoint</u>	Source line defined dynamic tracepoint: @peekq.c in QUECALLS.
<u>Minor Code</u>	32772 (0X8004)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code=%w Request=%w%d Address=%a Length=%d Element Code=%w Priority=%b

-----

## QUECALLS Major Code: 0X0016 Minor Code: 32773 (0X8005)

<u>Description</u>	(OS) DosPurgeQueue Post-Invocation
--------------------	------------------------------------

**Tracepoint** Source line defined dynamic tracepoint: @purgeq.c in QUECALLS.

**Minor Code** 32773 (0X8005)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Return Code=%w

-----

## QUECALLS Major Code: 0X0016 Minor Code: 32774 (0X8006)

**Description** (OS) DosQueryQueue Post-Invocation

**Tracepoint** Source line defined dynamic tracepoint: @queryq.c in QUECALLS.

**Minor Code** 32774 (0X8006)

**Trace Groups** No groups assigned.

**Trace Types** POST

**Traced Parameters**

Return Code=%w Number of Elements=%w

-----

## QUECALLS Major Code: 0X0016 Minor Code: 32775 (0X8007)

**Description** (OS) DosReadQueue Post-Invocation

**Tracepoint** Source line defined dynamic tracepoint: @readq.c in QUECALLS.

**Minor Code** 32775 (0X8007)

**Trace Groups** No groups assigned.

<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Return Code=%w Request=%w%d Address=%a Length=%d Priority=%b

---

## QUECALLS Major Code: 0X0016 Minor Code: 32776 (0X8008)

<b><u>Description</u></b>	(OS) DosWriteQueue Post-Invocation
<b><u>Tracepoint</u></b>	Source line defined dynamic tracepoint: @writeq.c in QUECALLS.
<b><u>Minor Code</u></b>	32776 (0X8008)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Return Code=%w

---

## SESMGR.DLL Trace Events

The tracepoints for the SESMGR.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for SESMGR Major Code: 0X0017, sorted by minor code.](#)  
[Trace events for SESMGR Major Code: 0X0017 ,sorted by tracepoint.](#)  
[Indirected Session Manager API Tracepoints.](#)

---

## Trace Events for SESMGR Major Code: 0X0017, Sorted by Minor Code

00001 (0X0001) (OS) DosSelectSession Pre-Invocation  
00002 (0X0002) (OS) DosSetSession Pre-Invocation  
00004 (0X0004) (OS) DosSMAAppNotify Pre-Invocation  
00005 (0X0005) (OS) DosSMChildExit Pre-Invocation  
00007 (0X0007) (OS) DosSMDoAppReq Pre-Invocation  
00008 (0X0008) (OS) DosSMFreeSGId Pre-Invocation  
00009 (0X0009) (OS) DosSMGetSGId Pre-Invocation  
00011 (0X000B) (OS) DosSMNotifyDD Pre-Invocation  
00012 (0X000C) (OS) DosSMNotifyDD2 Pre-Invocation  
00013 (0X000D) (OS) DosSMParentSwitch Pre-Invocation  
00014 (0X000E) (OS) DosSMMSGDoPopup Pre-Invocation  
00015 (0X000F) (OS) DosSMMSGEndPopup Pre-Invocation  
00016 (0X0010) (OS) DosSMMSGSet Pre-Invocation  
00017 (0X0011) (OS) DosSMMSGStart Pre-Invocation  
00018 (0X0012) (OS) DosSMMSGSwitch Pre-Invocation  
00019 (0X0013) (OS) DosSMMSGTerminate Pre-Invocation  
00020 (0X0014) (OS) DosStartSession Pre-Invocation  
00021 (0X0015) (OS) DosStopSession Pre-Invocation  
00023 (0X0017) (OS) ParentNotify Pre-Invocation  
00024 (0X0018) (OS) ParentSwitch Pre-Invocation  
00025 (0X0019) (OS) WriteTermQueue Pre-Invocation  
32769 (0X8001) (OS) DosSelectSession Post-Invocation  
32770 (0X8002) (OS) DosSetSession Post-Invocation  
32772 (0X8004) (OS) DosSMAAppNotify Post-Invocation  
32773 (0X8005) (OS) DosSMChildExit Post-Invocation  
32775 (0X8007) (OS) DosSMDoAppReq Post-Invocation  
32779 (0X800B) (OS) DosSMNotifyDD Post-Invocation  
32780 (0X800C) (OS) DosSMNotifyDD2 Post-Invocation  
32781 (0X800D) (OS) DosSMParentSwitch Post-Invocation  
32782 (0X800E) (OS) DosSMMSGDoPopup Post-Invocation  
32783 (0X800F) (OS) DosSMMSGEndPopup Post-Invocation  
32784 (0X8010) (OS) DosSMMSGSet Post-Invocation  
32785 (0X8011) (OS) DosSMMSGStart Post-Invocation  
32786 (0X8012) (OS) DosSMMSGSwitch Post-Invocation  
32787 (0X8013) (OS) DosSMMSGTerminate Post-Invocation  
32788 (0X8014) (OS) DosStartSession Post-Invocation  
32789 (0X8015) (OS) DosStopSession Post-Invocation  
32791 (0X8017) (OS) ParentNotify Post-Invocation  
32792 (0X8018) (OS) ParentSwitch Post-Invocation  
32793 (0X8019) (OS) WriteTermQueue Post-Invocation

-----

## Trace Events for SESMGR Major Code: 0X0017, Sorted by Tracepoint

DOSSELECTSESSION 00001 (0X0001)  
DOSSELECTSESSION\_POSTDT 32769 (0X8001)  
DOSSMADDSGQUEUE 00002 (0X0002)  
DOSSMADDSGQUEUE\_POSTDT 32770 (0X8002)  
DOSSMAPPNOTIFY 00004 (0X0004)  
DOSSMAPPNOTIFY\_POSTDT 32772 (0X8004)  
DOSSMDELSGQUEUE 00005 (0X0005)  
DOSSMDELSGQUEUE\_POSTDT 32773 (0X8005)  
DOSSMDOAPPREQ 00007 (0X0007)  
DOSSMFREESGID 00008 (0X0008)  
DOSSMGETSGQUEUE 00009 (0X0009)  
DOSSMGETSGQUEUE\_POSTDT 32775 (0X8007)  
DOSSMGETSTATUS 00021 (0X0015)  
DOSSMGETSTATUS\_POSTDT 32789 (0X8015)  
DOSSMNOTIFYDD 00011 (0X000B)  
DOSSMNOTIFYDD2 00012 (0X000C)  
DOSSMNOTIFYDD2\_POSTDT 32780 (0X800C)  
DOSSMNOTIFYDD\_POSTDT 32779 (0X800B)  
DOSSMPARENTSWITCH 00013 (0X000D)  
DOSSMPARENTSWITCH\_POSTDT 32781 (0X800D)

DOSSMSGDOPOPUP 00014 (0X000E)  
DOSSMSGDOPOPUP\_POSTDT 32782 (0X800E)  
DOSSMSGENDPOPUP 00015 (0X000F)  
DOSSMSGENDPOPUP\_POSTDT 32783 (0X800F)  
DOSSMSGSET 00016 (0X0010)  
DOSSMSGSET\_POSTDT 32784 (0X8010)  
DOSSMSGSTART 00017 (0X0011)  
DOSSMSGSTART\_POSTDT 32785 (0X8011)  
DOSSMSGSWITCH 00018 (0X0012)  
DOSSMSGSWITCH\_POSTDT 32786 (0X8012)  
DOSSMSGTERMINATE 00019 (0X0013)  
DOSSMSGTERMINATE\_POSTDT 32787 (0X8013)  
DOSSTARTSESSION 00020 (0X0014)  
DOSSTARTSESSION\_POSTDT 32788 (0X8014)  
PARENTNOTIFY 00023 (0X0017)  
PARENTNOTIFY\_POSTDT 32791 (0X8017)  
PARENTSWITCH 00024 (0X0018)  
PARENTSWITCH\_POSTDT 32792 (0X8018)  
WRITETERMQUEUE 00025 (0X0019)  
WRITETERMQUEUE\_POSTDT 32793 (0X8019)

## SESMGR Major Code: 0X0017 Minor Code: 1 (0X0001)

### Description

(OS) DosSelectSession Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSELECTSESSION

### Minor Code

1 (0X0001)

### Trace Groups

No groups assigned.

### Trace Types

PRE

### Traced Parameters

Reserved = %d

Select Option = %w

## SESMGR Major Code: 0X0017 Minor Code: 2 (0X0002)

### Description

(OS) DosSetSession Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMADDSGQUEUE

### Minor Code

2 (0X0002)

### Trace Groups

No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**  
  
Status Data - SetLength = %w  
SelectOpt = %w  
BondOption = %w  
Session Id = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 4 (0X0004)

**Description**  
(OS) DosSMAAppNotify Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: SESMGR.DOSSMAPPNOTIFY

**Minor Code**  
4 (0X0004)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**  
  
Sessions Return Code = %w  
Terminated Session Id = %w  
Notify Switch Action = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 5 (0X0005)

**Description**  
(OS) DosSMChildExit Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: SESMGR.DOSSMDELSGQUEUE

**Minor Code**  
5 (0X0005)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**

SGID = %w

-----

SESMGR Major Code: 0X0017 Minor Code: 7 (0X0007)

**Description**

(OS) DosSMDoAppReq Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMDOAPPREQ

**Minor Code**

7 (0X0007)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Request Header = %r%b

Request Data = %r%b

-----

SESMGR Major Code: 0X0017 Minor Code: 8 (0X0008)

**Description**

(OS) DosSMFreeSGId Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMFREESGID

**Minor Code**

8 (0X0008)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

SGID = %w

-----

SESMGR Major Code: 0X0017 Minor Code: 9 (0X0009)



<u>Description</u>	(OS) DosSMGetSGId Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSGQUEUE
<u>Minor Code</u>	9 (0X0009)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	None

## SESMGR Major Code: 0X0017 Minor Code: 11 (0X000B)

<u>Description</u>	(OS) DosSMNotifyDD Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD
<u>Minor Code</u>	11 (0X000B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	<div> <div>Outgoing SG</div> <div>= %w</div> </div> <div> <div>Incoming SG</div> <div>= %w</div> </div> <div> <div>Notification Type</div> <div>= %w</div> </div>

## SESMGR Major Code: 0X0017 Minor Code: 12 (0X000C)

<u>Description</u>	(OS) DosSMNotifyDD2 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD2
<u>Minor Code</u>	12 (0X000C)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**  
  
None

-----

## SESMGR Major Code: 0X0017 Minor Code: 13 (0X000D)

**Description**  
(OS) DosSMParentSwitch Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: SESMGR.DOSSMPARENTSWITCH

**Minor Code**  
13 (0X000D)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**  
  
Session Id               = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 14 (0X000E)

**Description**  
(OS) DosSMSGDoPopup Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGDOPOPUP

**Minor Code**  
14 (0X000E)

**Trace Groups**  
No groups assigned.

**Trace Types**  
PRE

**Traced Parameters**  
  
SGID                    = %w  
PID                     = %w

PopType           = %w  
Proc\_Type         = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 15 (0X000F)

### Description

(OS) DosSMSEndPopup Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGENDPOPUP

### Minor Code

15 (0X000F)

### Trace Groups

No groups assigned.

### Trace Types

PRE

### Traced Parameters

None

-----

## SESMGR Major Code: 0X0017 Minor Code: 16 (0X0010)

### Description

(OS) DosMSGSet Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSET

### Minor Code

16 (0X0010)

### Trace Groups

No groups assigned.

### Trace Types

PRE

### Traced Parameters

Bond Option       = %w  
Select Option     = %w  
Child Session Id   = %w  
Parent Session Id  = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 17 (0X0011)

**Description**

(OS) DosMSGStart Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSTART

**Minor Code**

17 (0X0011)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Reserved	= %d
Debug PID	= %w
Debug SID	= %w
PM Struct	= %d
Request Block	= %d
Environ Option	= %w
New Process Id	= %w
New Session	= %w
Parent Id	= %w
Program Inputs	= %r%b
Program Name	= %r%b
Program Title	= %s
Program Type	= %w
Asynchronous Option	= %w
Start Mode	= %w
Save Action	= %w

---

## SESMGR Major Code: 0X0017 Minor Code: 18 (0X0012)

**Description**

(OS) DosMSGSwitch Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSWITCH

**Minor Code**

18 (0X0012)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Reserved = %d

Debug SIG = %w

Session Id = %w

Switch Action = %w

---

## SESMGR Major Code: 0X0017 Minor Code: 19 (0X0013)

**Description**

(OS) DosMSGSTerminate Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSTERMINATE

**Minor Code**

19 (0X0013)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Reserved = %d

Session Id = %w

---

## SESMGR Major Code: 0X0017 Minor Code: 20 (0X0014)

**Description**

(OS) DosStartSession Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSTARTSESSION

**Minor Code**

20 (0X0014)

**Trace Groups**

No groups assigned.

**Trace Types**

PRE

**Traced Parameters**

Start Data = %r%w

-----

## SESMGR Major Code: 0X0017 Minor Code: 21 (0X0015)

### Description

(OS) DosStopSession Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSTATUS

### Minor Code

21 (0X0015)

### Trace Groups

No groups assigned.

### Trace Types

PRE

### Traced Parameters

Reserved = %d

Session Id = %w

Stop Session Option = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 23 (0X0017)

### Description

(OS) ParentNotify Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.PARENTNOTIFY

### Minor Code

23 (0X0017)

### Trace Groups

No groups assigned.

### Trace Types

PRE

### Traced Parameters

Session Id = %w

TermType = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 24 (0X0018)

<b>Description</b>	(OS) ParentSwitch Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: SESMGR.PARENTSWITCH
<b>Minor Code</b>	24 (0X0018)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	
	Session Id = %w

## SESMGR Major Code: 0X0017 Minor Code: 25 (0X0019)

<b>Description</b>	(OS) WriteTermQueue Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: SESMGR.WRITETERMQUEUE
<b>Minor Code</b>	25 (0X0019)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	
	Action = %w
	Parent Process Id = %w
	Parent Screen Group = %w
	Terminating Session SID = %w
	PID = %w

## SESMGR Major Code: 0X0017 Minor Code: 32769 (0X8001)

<b>Description</b>	(OS) DosSelectSession Post-Invocation
--------------------	---------------------------------------

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: SESMGR.DOSSELECTSESSION_POSTDT
<b><u>Minor Code</u></b>	32769 (0X8001)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
	Return Code = %w

## SESMGR Major Code: 0X0017 Minor Code: 32770 (0X8002)

<b><u>Description</u></b>	(OS) DosSetSession Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMADDSGQUEUE_POSTDT
<b><u>Minor Code</u></b>	32770 (0X8002)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	
	Return Code = %w

## SESMGR Major Code: 0X0017 Minor Code: 32772 (0X8004)

<b><u>Description</u></b>	(OS) DosSMAppNotify Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMAPPNOTIFY_POSTDT
<b><u>Minor Code</u></b>	32772 (0X8004)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	



Return Code = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 32773 (0X8005)

### Description

(OS) DosSMChildExit Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMDELSGQUEUE\_POSTDT

### Minor Code

32773 (0X8005)

### Trace Groups

No groups assigned.

### Trace Types

POST

### Traced Parameters

Return Code = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 32775 (0X8007)

### Description

(OS) DosSMDoAppReq Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSGQUEUE\_POSTDT

### Minor Code

32775 (0X8007)

### Trace Groups

No groups assigned.

### Trace Types

POST

### Traced Parameters

Return Code = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 32779 (0X800B)

### Description

(OS) DosSMNotifyDD Post-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD_POSTDT
<u>Minor Code</u>	32779 (0X800B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

## SESMGR Major Code: 0X0017 Minor Code: 32780 (0X800C)

<u>Description</u>	(OS) DosSMNotifyDD2 Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMNOTIFYDD2_POSTDT
<u>Minor Code</u>	32780 (0X800C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	DosDevIOCTLsRC = %w

## SESMGR Major Code: 0X0017 Minor Code: 32781 (0X800D)

<u>Description</u>	(OS) DosSMParentSwitch Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMPARENTSWITCH_POSTDT
<u>Minor Code</u>	32781 (0X800D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	

Switched = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 32782 (0X800E)

### Description

(OS) DosMSGDoPopup Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGDOPOPUP\_POSTDT

### Minor Code

32782 (0X800E)

### Trace Groups

No groups assigned.

### Trace Types

POST

### Traced Parameters

Return Code = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 32783 (0X800F)

### Description

(OS) DosMSGEndPopup Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGENDPOPUP\_POSTDT

### Minor Code

32783 (0X800F)

### Trace Groups

No groups assigned.

### Trace Types

POST

### Traced Parameters

Return Code = %w

-----

## SESMGR Major Code: 0X0017 Minor Code: 32784 (0X8010)

### Description

(OS) DosMSGSet Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSET\_POSTDT

**Minor Code**

32784 (0X8010)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

**SESMGR Major Code: 0X0017 Minor Code: 32785 (0X8011)**

**Description**

(OS) DosMSGStart Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSTART\_POSTDT

**Minor Code**

32785 (0X8011)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

**SESMGR Major Code: 0X0017 Minor Code: 32786 (0X8012)**

**Description**

(OS) DosMSGSwitch Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGSWITCH\_POSTDT

**Minor Code**

32786 (0X8012)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code           = %w  
  
-----

SESMGR Major Code: 0X0017 Minor Code: 32787 (0X8013)

**Description**

(OS) DosMSGTerminate Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSMSGTERMINATE\_POSTDT

**Minor Code**

32787 (0X8013)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code           = %w  
  
-----

SESMGR Major Code: 0X0017 Minor Code: 32788 (0X8014)

**Description**

(OS) DosStartSession Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.DOSSTARTSESSION\_POSTDT

**Minor Code**

32788 (0X8014)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

New Process Id       = %w  
  
New Session           = %w  
  
Return Code           = %w  
  
-----

SESMGR Major Code: 0X0017 Minor Code: 32789 (0X8015)

<u>Description</u>	(OS) DosStopSession Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.DOSSMGETSTATUS_POSTDT
<u>Minor Code</u>	32789 (0X8015)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

## SESMGR Major Code: 0X0017 Minor Code: 32791 (0X8017)

<u>Description</u>	(OS) ParentNotify Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.PARENTNOTIFY_POSTDT
<u>Minor Code</u>	32791 (0X8017)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Switched = %w

## SESMGR Major Code: 0X0017 Minor Code: 32792 (0X8018)

<u>Description</u>	(OS) ParentSwitch Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: SESMGR.PARENTSWITCH_POSTDT
<u>Minor Code</u>	32792 (0X8018)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Switched = %w

SESMGR Major Code: 0X0017 Minor Code: 32793 (0X8019)

**Description**

(OS) WriteTermQueue Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: SESMGR.WRITETERMQUEUE\_POSTDT

**Minor Code**

32793 (0X8019)

**Trace Groups**

No groups assigned.

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

Indirected Session Manager API Tracepoints

The following table lists 32-bit pre-invocation tracepoints for SESMGR.DLL APIs that are indirected via SESMGR thunking layer. These should be trace in conjunction with their corresponding SESMGR 16-bit API tracepoints.

<i>SESMGR API</i>	<i>Minor code</i>	<i>Types</i>
DOS32SELECTSESSION	32	PRE , API
DOS32SETSESSION	33	PRE , API
DOS32STARTSESSION	34	PRE , API
DOS32STOPSESSION	35	PRE , API

Multi-Media Extensions

The tracepoints for the Multi-Media Extensions major code are identified in the following tables. These tracepoints are static tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for Multi-Media Extensions major code: 0X006D, sorted by minor code.  
Trace events for Multi-Media Extensions major code: 0X006D ,sorted by tracepoint.

-----

## Trace Events for Multi-Media Extensions Major Code: 0X006D, Sorted by Minor Code

00001 (0X0001) SSM\_ShcAssociate\_Entry  
00002 (0X0002) SSM\_ShcClose\_Entry  
00003 (0X0003) SSM\_ShcCreate\_Entry  
00004 (0X0004) SSM\_ShcDestroy\_Entry  
00005 (0X0005) SSM\_ShcStart\_Entry  
00006 (0X0006) SSM\_ShcStop\_Entry  
00007 (0X0007) SSM\_ShcSeek\_Entry  
00008 (0X0008) SSM\_ShcEnableEvt\_Entry  
00009 (0X0009) SSM\_ShcDisableEvt\_Entry  
00010 (0X000A) SSM\_ShcEnableSync\_Entry  
00011 (0X000B) SSM\_ShcDisableSync\_Entry  
00012 (0X000C) SSM\_ShcGetTime\_Entry  
00013 (0X000D) SSM\_ShcGetProtocol\_Entry  
00014 (0X000E) SSM\_ShcInstProtocol\_Entry  
00015 (0X000F) SSM\_ShcEnumProtocol\_Entry  
00016 (0X0010) SSM\_ShcNegotReslt\_Entry  
00017 (0X0011) SSM\_ShcSendMsg\_Entry  
00018 (0X0012) UNUSED\_HOOK\_109\_018  
00019 (0X0013) UNUSED\_HOOK\_109\_019  
00020 (0X0014) SSM\_ProcRun\_Entry  
00021 (0X0015) SSM\_ProcBlock\_Entry  
00022 (0X0016) SSM\_loctl\_Entry  
00023 (0X0017) SSM\_IDC\_Call\_To\_SHC  
00024 (0X0018) SSM\_IDCNotifyEntry  
00025 (0X0019) SSM\_IDCDeRegEntry  
00026 (0X001A) SSM\_IDCRptEvtEntry  
00027 (0X001B) GET\_EMPTY\_Entry  
00028 (0X001C) GET\_FULL\_Entry  
00029 (0X001D) RET\_EMPTY\_Entry  
00030 (0X001E) RET\_FULL\_Entry  
00031 (0X001F) BUF\_Record\_Entry  
00032 (0X0020) SMH\_ReportEvent0  
00033 (0X0021) UNUSED\_HOOK\_109\_033  
00034 (0X0022) UNUSED\_HOOK\_109\_034  
00035 (0X0023) SSM\_SmhRing3\_Entry  
00036 (0X0024) SSM\_SmhRegister\_Entry  
00037 (0X0025) SSM\_SmhReportEvtnt\_Entry  
00038 (0X0026) SSM\_SmhNotify\_Entry  
00039 (0X0027) SSM\_SmhDeRegister\_Entry  
00040 (0X0028) SSM\_SmhLockMem\_Entry  
00041 (0X0029) SSM\_SpiAssociate\_Entry  
00042 (0X002A) SSM\_SpiCreate\_Entry  
00043 (0X002B) SSM\_SpiDestroy\_Entry  
00044 (0X002C) SSM\_SpiEnumProtocol\_Entry  
00045 (0X002D) SSM\_SpiEnumHndlr\_Entry  
00046 (0X002E) SSM\_SpiGetHndlr\_Entry  
00047 (0X002F) SSM\_SpiGetProtocol\_Entry  
00048 (0X0030) SSM\_SpiGetTime\_Entry  
00049 (0X0031) SSM\_SpiInstProtocol\_Entry  
00050 (0X0032) SSM\_SpiSeekStrm\_Entry  
00051 (0X0033) SSM\_SpiStartStrm\_Entry  
00052 (0X0034) SSM\_SpiStopStrm\_Entry  
00053 (0X0035) SSM\_SpiEnableEvent\_Entry



00054 (0X0036) SSM\_SpiDisableEvent\_Entry  
00055 (0X0037) SSM\_SpiEnableSync\_Entry  
00056 (0X0038) SSM\_SpiDisableSync\_Entry  
00057 (0X0039) SSM\_ShcRouter\_Entry  
00058 (0X003A) SSM\_ShcRouterRing3\_Entry  
00059 (0X003B) SSM\_SmhRptEventSync\_Entry  
00060 (0X003C) SSM\_AppEvent3\_Entry  
00061 (0X003D) SSM\_AppEventQ\_Entry  
00062 (0X003E) SSM\_SmhNotifyIOCtl\_Entry  
00063 (0X003F) SSM\_QueueEventIOCtl\_Entry  
00064 (0X0040) SSM\_CreateBDS  
00065 (0X0041) SSM\_SpiDetSyncMaster\_Entry  
00066 (0X0042) SSM\_SpiSendMsg\_Entry  
00067 (0X0043) SSM\_SpiAssStrNetEntry  
00068 (0X0044) SSM\_SpiCloseStrNetEntry  
00069 (0X0045) SSM\_SpiConnStrEntry  
00070 (0X0046) SSM\_SpiDeactStrEntry  
00071 (0X0047) SSM\_SpiAddStrConnEntry  
00072 (0X0048) SSM\_SpiOpenStrNetEntry  
00073 (0X0049) SSM\_SpiReactStrEntry  
00074 (0X004A) SSM\_SpiRemStrConnEntry  
00075 (0X004B) SSM\_SpiSeekStrNetEntry  
00076 (0X004C) SSM\_SpiStartStrNetEntry  
00077 (0X004D) SSM\_SpiStopStrNetEntry  
00078 ( 0X004E) SSM\_SpiUnassStrNetEntry  
00080 (0X0050) ACP\_ACPADevintEntry  
00081 (0X0051) ADH\_ADShIntHandlerEntry  
00082 (0X0052) ADH\_SHCCreateEntry  
00083 (0X0053) ADH\_SHCStartEntry  
00084 (0X0054) ADH\_SHCStopEntry  
00085 (0X0055) ADH\_GetEmptyEntry  
00086 (0X0056) ADH\_ReturnFullEntry  
00087 (0X0057) ADH\_GetFullEntry  
00088 (0X0058) ADH\_ReturnEmptyEntry  
00089 (0X0059) ADH\_ReportEOSEEntry  
00090 (0X005A) ADH\_ReportErrorEntry  
00091 (0X005B) ADH\_ReportPLCueEntry  
00092 (0X005C) ADH\_ReportCueTimeEntry  
00093 (0X005D) ADH\_ReportCueDataEntry  
00094 (0X005E) ADH\_ReportDataUnderEntry  
00095 (0X005F) ADH\_ReportSyncOverEntry  
00096 (0X0060) ADH\_EventHandlerEntry  
00097 (0X0061) ACP\_DevReportError  
00101 (0X0065) ADSh\_AddBufEntry  
00102 (0X0066) ADSh\_RemoveBufEntry  
00103 (0X0067) ADSh\_IDCWriteEntry  
00104 (0X0068) ADSh\_GetBufferEntry  
00105 (0X0069) ADSh\_RetBufferEntry  
00106 (0X006A) ADSh\_DLLIntEntry  
00107 (0X006B) ADSh\_IDCControlEntry  
00108 (0X006C) ADSh\_IDCSetupEntry  
00109 (0X006D) ADSh\_SendBufferEntry  
00112 (0X0070) CDSH\_StartCdReadIoctl  
00113 (0X0071) CDSH\_StartCddaRead  
00114 (0X0072) CDSH\_StartCddaWrite  
00129 (0X0081) SSM\_ShcAssociate\_Exit  
00130 (0X0082) SSM\_ShcClose\_Exit  
00131 (0X0083) SSM\_ShcCreate\_Exit  
00132 (0X0084) SSM\_ShcDestroy\_Exit  
00133 (0X0085) SSM\_ShcStart\_Exit  
00134 (0X0086) SSM\_ShcStop\_Exit  
00135 (0X0087) SSM\_ShcSeek\_Exit  
00136 (0X0088) SSM\_ShcEnableEvt\_Exit  
00137 (0X0089) SSM\_ShcDisableEvt\_Exit  
00138 (0X008A) SSM\_ShcEnableSync\_Exit  
00139 (0X008B) SSM\_ShcDisableSync\_Exit  
00140 (0X008C) SSM\_ShcGetTime\_Exit  
00141 (0X008D) SSM\_ShcGetProtocol\_Exit  
00142 (0X008E) SSM\_ShcInstProtocol\_Exit  
00143 (0X008F) SSM\_ShcEnumProtocol\_Exit  
00144 (0X0090) SSM\_ShcNegotReslt\_Exit  
00145 (0X0091) SSM\_ShcSendMsg\_Exit  
00146 (0X0092) UNUSED\_HOOK\_109\_146

00147 (0X0093) UNUSED\_HOOK\_109\_147  
00148 (0X0094) SSM\_ProcRun\_Exit  
00149 (0X0095) SSM\_ProcBlock\_Exit  
00150 (0X0096) SSM\_loctl\_Exit  
00151 (0X0097) SSM\_IDC\_Ret\_From\_SHC  
00152 (0X0098) SSM\_IDCNotifyExit  
00153 (0X0099) SSM\_IDCDeRegExit  
00154 (0X009A) SSM\_IDCRptEvtExit  
00155 (0X009B) GET\_EMPTY\_Exit  
00156 (0X009C) GET\_FULL\_Exit  
00157 (0X009D) RET\_EMPTY\_Exit  
00158 (0X009E) RET\_FULL\_Exit  
00159 (0X009F) BUF\_Record\_Exit  
00160 (0X00A0) SSM\_Negotiate\_error  
00161 (0X00A1) UNUSED\_HOOK\_109\_161  
00162 (0X00A2) UNUSED\_HOOK\_109\_162  
00163 (0X00A3) SSM\_SmhRing3\_Exit  
00164 (0X00A4) SSM\_SmhRegister\_Exit  
00165 (0X00A5) SSM\_SmhReportEvt\_Exit  
00166 (0X00A6) SSM\_SmhNotify\_Exit  
00167 (0X00A7) SSM\_SmhDeRegister\_Exit  
00168 (0X00A8) SSM\_SmhLockMem\_Exit  
00169 (0X00A9) SSM\_SpiAssociate\_Exit  
00170 (0X00AA) SSM\_SpiCreate\_Exit  
00171 (0X00AB) SSM\_SpiDestroy\_Exit  
00172 (0X00AC) SSM\_SpiEnumProtocol\_Exit  
00173 (0X00AD) SSM\_SpiEnumHndlr\_Exit  
00174 ( 0X00AE) SSM\_SpiGetHndlr\_Exit  
00175 (0X00AF) SSM\_SpiGetProtocol\_Exit  
00176 (0X00B0) SSM\_SpiGetTime\_Exit  
00177 (0X00B1) SSM\_SpiInstProtocol\_Exit  
00178 (0X00B2) SSM\_SpiSeekStrm\_Exit  
00179 (0X00B3) SSM\_SpiStartStrm\_Exit  
00180 (0X00B4) SSM\_SpiStopStrm\_Exit  
00181 (0X00B5) SSM\_SpiEnableEvent\_Exit  
00182 (0X00B6) SSM\_SpiDisableEvent\_Exit  
00183 (0X00B7) SSM\_SpiEnableSync\_Exit  
00184 (0X00B8) SSM\_SpiDisableSync\_Exit  
00185 (0X00B9) SSM\_ShcRouter\_Exit  
00186 (0X00BA) SSM\_ShcRouterRing3\_Exit  
00187 (0X00BB) SSM\_xxxUnused1  
00188 (0X00BC) SSM\_AppEvent3\_Exit  
00189 (0X00BD) SSM\_AppEventQ\_Exit  
00190 (0X00BE) SSM\_SmhNotifyIOctl\_Exit  
00191 (0X00BF) SSM\_QueueEventIOctl\_Exit  
00192 (0X00C0) SSM\_CreateBDSNotMax  
00193 (0X00C1) SSM\_SpiDetSyncMaster\_Exit  
00194 (0X00C2) SSM\_SpiSendMsg\_Exit  
00195 (0X00C3) SSM\_SpiAssStrNetExit  
00196 (0X00C4) SSM\_SpiCloseStrNetExit  
00197 (0X00C5) SSM\_SpiConnStrExit  
00198 (0X00C6) SSM\_SpiDeactStrExit  
00199 (0X00C7) SSM\_SpiAddStrConnExit  
00200 (0X00C8) SSM\_SpiOpenStrNetExit  
00201 (0X00C9) SSM\_SpiReactStrExit  
00202 (0X00CA) SSM\_SpiRemStrConnExit  
00203 (0X00CB) SSM\_SpiSeekStrNetExit  
00204 (0X00CC) SSM\_SpiStartStrNetExit  
00205 (0X00CD) SSM\_SpiStopStrNetExit  
00206 (0X00CE) SSM\_SpiUnassStrNetExit  
00208 (0X00D0) ACP\_ACPADevIntExit  
00209 (0X00D1) ADH\_ADShIntHandlerExit  
00210 (0X00D2) ADH\_SHCCreateExit  
00211 (0X00D3) ADH\_SHCStartExit  
00212 (0X00D4) ADH\_SHCStopExit  
00213 (0X00D5) ADH\_GetEmptyExit  
00214 (0X00D6) ADH\_ReturnFullExit  
00215 (0X00D7) ADH\_GetFullExit  
00216 (0X00D8) ADH\_ReturnEmptyExit  
00217 (0X00D9) ADH\_ReportEOSExit  
00218 (0X00DA) ADH\_ReportErrorExit  
00219 (0X00DB) ADH\_ReportPLCueExit  
00220 (0X00DC) ADH\_ReportCueTimeExit

00221 (0X00DD) ADH\_ReportCueDataExit  
00222 (0X00DE) ADH\_ReportDataUnderExit  
00223 (0X00DF) ADH\_ReportSyncOverExit  
00224 (0X00E0) ADH\_EventHandlerExit  
00225 (0X00E1) ACP\_DevReportError  
00229 (0X00E5) ADSH\_AddBufExit  
00230 (0X00E6) ADSH\_RemoveBufExit  
00231 (0X00E7) ADSH\_IDCWriteExit  
00232 (0X00E8) ADSH\_GetBufferExit  
00233 (0X00E9) ADSH\_RetBufferExit  
00234 (0X00EA) ADSH\_DLLIntExit  
00235 (0X00EB) ADSH\_IDCControlExit  
00236 (0X00EC) ADSH\_IDCSetupExit  
00237 (0X00ED) ADSH\_SendBufferExit  
00240 (0X00F0) CDSH\_StopCdReadIoctl  
00241 (0X00F1) CDSH\_StopCddaRead  
00242 (0X00F2) CDSH\_StopCddaWrite  
00257 (0X0101) MDM\_mciSendStringEntry  
00258 (0X0102) MDM\_mciSendCommandEntry  
00259 (0X0103) MDM\_mciOpen\_Entry  
00260 (0X0104) MDM\_MCDOpen\_Entry  
00261 (0X0105) MDM\_mciMakeActive\_Entry  
00262 (0X0106) MDM\_mciClose\_Entry  
00263 (0X0107) MDM\_RESTORE\_Entry  
00264 (0X0108) MDM\_SAVE\_Entry  
00265 (0X0109) MDM\_Thread\_call\_Entry  
00324 (0X0144) MTSH\_GetEmptyEntry  
00325 (0X0145) MTSH\_ReturnFullEntry  
00385 (0X0181) MDM\_mciSendStringExit  
00386 (0X0182) MDM\_mciSendCommandExit  
00387 (0X0183) MDM\_mciOpen\_Exit  
00388 (0X0184) MDM\_MCDOpen\_Exit  
00389 ( 0X0185) MDM\_mciMakeActive\_Exit  
00390 (0X0186) MDM\_mciClose\_Exit  
00391 (0X0187) MDM\_RESTORE\_EXIT  
00392 (0X0188) MDM\_SAVE\_EXIT  
00393 (0X0189) MDM\_Thread\_call\_Exit  
00452 (0X01C4) MTSH\_GetEmptyExit  
00453 (0X01C5) MTSH\_ReturnFullExit  
00513 (0X0201) MIO\_DosOpen\_Entry  
00514 (0X0202) MIO\_DosRead\_Entry  
00515 (0X0203) MIO\_DosWrite\_Entry  
00516 (0X0204) MIO\_DosSeek\_Entry  
00517 (0X0205) MIO\_DosClose\_Entry  
00518 (0X0206) MIO\_DosDelete\_Entry  
00519 (0X0207) MIO\_MemOpen\_Entry  
00520 (0X0208) MIO\_MemRead\_Entry  
00521 (0X0209) MIO\_MemWrite\_Entry  
00522 (0X020A) MIO\_MemSeek\_Entry  
00523 (0X020B) MIO\_MemClose\_Entry  
00524 (0X020C) MIO\_mmioOpen\_Entry  
00525 (0X020D) MIO\_mmioClose\_Entry  
00526 (0X020E) MIO\_mmioRead\_Entry  
00527 (0X020F) MIO\_mmioWrite\_Entry  
00528 (0X0210) MIO\_mmioSeek\_Entry  
00529 (0X0211) MIO\_mmioFlush\_Entry  
00530 (0X0212) MIO\_mmioAscend\_Entry  
00531 (0X0213) MIO\_mmioDescend\_Entry  
00532 (0X0214) MIO\_mmioAdvance\_Entry  
00533 (0X0215) MIO\_mmioInstIOProc\_Entry  
00534 (0X0216) MIO\_mmioSendMsg\_Entry  
00535 (0X0217) MIO\_mmioAquireSem\_Entry  
00536 (0X0218) MIO\_mmioDiscardSem\_Entry  
00537 (0X0219) MIO\_mmioCreateChunk\_Entry  
00538 (0X021A) MIO\_mmioCFOpen\_Entry  
00539 (0X021B) MIO\_mmioCFClose\_Entry  
00540 (0X021C) MIO\_mmioCFRmvShrEnt\_Entry  
00541 (0X021D) MIO\_mmioCFAddShrEnt\_Entry  
00542 (0X021E) MIO\_mmioCFSetInfo\_Entry  
00543 (0X021F) MIO\_mmioCFGetInfo\_Entry  
00544 (0X0220) MIO\_mmioCFOpnTmpElem\_Entry  
00545 (0X0221) MIO\_mmioCFClstmpElem\_Entry  
00546 (0X0222) MIO\_mmioCFAddEnt\_Entry

00547 (0X0223) MIO\_mmioCFChgEnt\_Entry  
00548 (0X0224) MIO\_mmioCFCopy\_Entry  
00549 (0X0225) MIO\_mmioCFDelEnt\_Entry  
00550 (0X0226) MIO\_mmioCFAddElem\_Entry  
00551 (0X0227) MIO\_mmioCFFndEnt\_Entry  
00552 (0X0228) MIO\_mmiolIdentFile\_Entry  
00553 (0X0229) MIO\_MidiOpen\_Entry  
00554 (0X022A) MIO\_MidiRead\_Entry  
00555 (0X022B) MIO\_MidiWrite\_Entry  
00556 (0X022C) MIO\_MidiSeek\_Entry  
00557 (0X022D) MIO\_MidiClose\_Entry  
00576 (0X0240) MSH\_RdPlayList\_Entry  
00577 (0X0241) MSH\_WrPlayList\_Entry  
00592 (0X0250) SWVR\_GetImageEntry  
00593 (0X0251) SWVR\_ComplIdxFrameEntry  
00594 (0X0252) SWVR\_CompRefrFrameEntry  
00595 (0X0253) SWVR\_PostBufferFullEntry  
00596 (0X0254) SWVR\_PostBufferEmptyEntry  
00597 (0X0255) SWVR\_WaitMsgBufferFullEntry  
00598 (0X0256) SWVR\_WaitMsgBufferEmptyEntry  
00599 (0X0257) SWVR\_PostCountErrorEntry  
00600 (0X0258) SWVR\_BufferWriteStartEntry  
00601 (0X0259) SWVR\_VCADosDevIOCTLEntry  
00602 (0X025A) SWVR\_VCABufferCopyEntry  
00603 (0X025B) SWVR\_VCAtoX4CopyEntry  
00604 (0X025C) SWVR\_VCAtoX2CopyEntry  
00605 (0X025D) SWVR\_VCA640CopyEntry  
00612 (0X0264) SVSH\_DecompressBufferEntry  
00613 (0X0265) ULDC\_DLLDroppedEntry  
00617 (0X0269) SVMC\_SetBitmapEntry  
00618 (0X026A) SVMC\_BitBltEntry  
00619 (0X026B) SVMC\_BlittUpEntry  
00620 (0X026C) SVMC\_SpiMoveCoorEntry  
00621 (0X026D)  
00641 (0X0281) MIO\_DosOpen\_Exit  
00642 (0X0282) MIO\_DosRead\_Exit  
00643 (0X0283) MIO\_DosWrite\_Exit  
00644 ( 0X0284) MIO\_DosSeek\_Exit  
00645 (0X0285) MIO\_DosClose\_Exit  
00646 (0X0286) MIO\_DosDelete\_Exit  
00647 (0X0287) MIO\_MemOpen\_Exit  
00648 (0X0288) MIO\_MemRead\_Exit  
00649 (0X0289) MIO\_MemWrite\_Exit  
00650 (0X028A) MIO\_MemSeek\_Exit  
00651 (0X028B) MIO\_MemClose\_Exit  
00652 (0X028C) MIO\_mmioOpen\_Exit  
00653 (0X028D) MIO\_mmioClose\_Exit  
00654 (0X028E) MIO\_mmioRead\_Exit  
00655 (0X028F) MIO\_mmioWrite\_Exit  
00656 (0X0290) MIO\_mmioSeek\_Exit  
00657 (0X0291) MIO\_mmioFlush\_Exit  
00658 (0X0292) MIO\_mmioAscend\_Exit  
00659 (0X0293) MIO\_mmioDescend\_Exit  
00660 (0X0294) MIO\_mmioAdvance\_Exit  
00661 (0X0295) MIO\_mmiolnstlIOProc\_Exit  
00662 (0X0296) MIO\_mmioSendMsg\_Exit  
00663 (0X0297) MIO\_mmioAquireSem\_Exit  
00664 (0X0298) MIO\_mmioDiscardSem\_Exit  
00665 (0X0299) MIO\_mmioCreateChunk\_Exit  
00666 (0X029A) MIO\_mmioCFOpen\_Exit  
00667 (0X029B) MIO\_mmioCFClose\_Exit  
00668 (0X029C) MIO\_mmioCFRmvShrEnt\_Exit  
00669 (0X029D) MIO\_mmioCFAddShrEnt\_Exit  
00670 (0X029E) MIO\_mmioCFSetInfo\_Exit  
00671 (0X029F) MIO\_mmioCFGetInfo\_Exit  
00672 (0X02A0) MIO\_mmioCFOpnTmpElem\_Exit  
00673 (0X02A1) MIO\_mmioCFClstmpElem\_Exit  
00674 (0X02A2) MIO\_mmioCFAddEnt\_Exit  
00675 (0X02A3) MIO\_mmioCFChgEnt\_Exit  
00676 (0X02A4) MIO\_mmioCFCopy\_Exit  
00677 (0X02A5) MIO\_mmioCFDelEnt\_Exit  
00678 (0X02A6) MIO\_mmioCFAddElem\_Exit  
00679 (0X02A7) MIO\_mmioCFFndEnt\_Exit

00680 (0X02A8) MIO\_mmiIdentFile\_Exit  
00681 (0X02A9) MIO\_MidiOpen\_Exit  
00682 (0X02AA) MIO\_MidiRead\_Exit  
00683 (0X02AB) MIO\_MidiWrite\_Exit  
00684 (0X02AC) MIO\_MidiSeek\_Exit  
00685 (0X02AD) MIO\_MidiClose\_Exit  
00704 (0X02C0) MSH\_RdPlayList\_Exit  
00705 (0X02C1) MSH\_WrPlayList\_Exit  
00720 (0X02D0) SWVR\_GetImageExit  
00721 (0X02D1) SWVR\_ComplIdxFrameExit  
00722 (0X02D2) SWVR\_CompRefrFrameExit  
00723 (0X02D3) SWVR\_PostBufferFullExit  
00724 (0X02D4) SWVR\_PostBufferEmptyExit  
00725 (0X02D5) SWVR\_WaitMsgBufferFullExit  
00726 (0X02D6) SWVR\_WaitMsgBufferEmptyExit  
00727 (0X02D7) SWVR\_PostCountErrorExit  
00728 (0X02D8) SWVR\_BufferWriteDoneExit  
00729 (0X02D9) SWVR\_VCADosDevIOCTLExit  
00730 (0X02DA) SWVR\_VCABufferCopyExit  
00731 (0X02DB) SWVR\_VCAtoX4CopyExit  
00732 (0X02DC) SWVR\_VCAtoX2CopyExit  
00733 (0X02DD) SWVR\_VCA640CopyExit  
00740 (0X02E4) SVSH\_DecompBufferExit  
00741 (0X02E5) ULDC\_DLLDroppedExit  
00745 (0X02E9) SVMC\_SetBitmapExit  
00746 (0X02EA) SVMC\_BitBitExit  
00747 (0X02EB) SVMC\_BlitUpExit  
00748 (0X02EC) SVMC\_SpiMoveCoorExit

---

## Trace Events for Multi-Media Extensions Major Code: 0X006D, Sorted by Tracepoint

00621 (0X026D)  
ACP\_ACPADevIntEntry 00080 (0X0050)  
ACP\_ACPADevIntExit 00208 (0X00D0)  
ACP\_DevReportError 00097 (0X0061)  
ACP\_DevReportError 00225 (0X00E1)  
ADH\_ADShIntHandlerEntry 00081 (0X0051)  
ADH\_ADShIntHandlerExit 00209 (0X00D1)  
ADH\_EventHandlerEntry 00096 (0X0060)  
ADH\_EventHandlerExit 00224 (0X00E0)  
ADH\_GetEmptyEntry 00085 (0X0055)  
ADH\_GetEmptyExit 00213 (0X00D5)  
ADH\_GetFullEntry 00087 (0X0057)  
ADH\_GetFullExit 00215 (0X00D7)  
ADH\_ReportCueDataEntry 00093 (0X005D)  
ADH\_ReportCueDataExit 00221 (0X00DD)  
ADH\_ReportCueTimeEntry 00092 (0X005C)  
ADH\_ReportCueTimeExit 00220 (0X00DC)  
ADH\_ReportDataUnderEntry 00094 (0X005E)  
ADH\_ReportDataUnderExit 00222 (0X00DE)  
ADH\_ReportEOSEEntry 00089 (0X0059)  
ADH\_ReportEOSEExit 00217 (0X00D9)  
ADH\_ReportErrorEntry 00090 (0X005A)  
ADH\_ReportErrorExit 00218 (0X00DA)  
ADH\_ReportPLCueEntry 00091 (0X005B)  
ADH\_ReportPLCueExit 00219 (0X00DB)  
ADH\_ReportSyncOverEntry 00095 (0X005F)  
ADH\_ReportSyncOverExit 00223 (0X00DF)  
ADH\_ReturnEmptyEntry 00088 (0X0058)  
ADH\_ReturnEmptyExit 00216 (0X00D8)  
ADH\_ReturnFullEntry 00086 (0X0056)  
ADH\_ReturnFullExit 00214 (0X00D6)  
ADH\_SHCCreateEntry 00082 (0X0052)

ADH\_SHCCreateExit 00210 (0X00D2)  
ADH\_SHCStartEntry 00083 (0X0053)  
ADH\_SHCStartExit 00211 (0X00D3)  
ADH\_SHCStopEntry 00084 (0X0054)  
ADH\_SHCStopExit 00212 (0X00D4)  
ADSH\_AddBufEntry 00101 (0X0065)  
ADSH\_AddBufExit 00229 (0X00E5)  
ADSH\_DLLIntEntry 00106 (0X006A)  
ADSH\_DLLIntExit 00234 (0X00EA)  
ADSH\_GetBufferEntry 00104 (0X0068)  
ADSH\_GetBufferExit 00232 (0X00E8)  
ADSH\_IDCControlEntry 00107 (0X006B)  
ADSH\_IDCControlExit 00235 (0X00EB)  
ADSH\_IDCSetupEntry 00108 (0X006C)  
ADSH\_IDCSetupExit 00236 (0X00EC)  
ADSH\_IDCWriteEntry 00103 (0X0067)  
ADSH\_IDCWriteExit 00231 (0X00E7)  
ADSH\_RemoveBufEntry 00102 (0X0066)  
ADSH\_RemoveBufExit 00230 (0X00E6)  
ADSH\_RetBufferEntry 00105 (0X0069)  
ADSH\_RetBufferExit 00233 (0X00E9)  
ADSH\_SendBufferEntry 00109 (0X006D)  
ADSH\_SendBufferExit 00237 (0X00ED)  
BUF\_Record\_Entry 00031 (0X001F)  
BUF\_Record\_Exit 00159 (0X009F)  
CDSH\_StartCddaRead 00113 (0X0071)  
CDSH\_StartCddaWrite 00114 (0X0072)  
CDSH\_StartCdReadIoctl 00112 (0X0070)  
CDSH\_StopCddaRead 00241 (0X00F1)  
CDSH\_StopCddaWrite 00242 (0X00F2)  
CDSH\_StopCdReadIoctl 00240 (0X00F0)  
GET\_EMPTY\_Entry 00027 (0X001B)  
GET\_EMPTY\_Exit 00155 (0X009B)  
GET\_FULL\_Entry 00028 (0X001C)  
GET\_FULL\_Exit 00156 (0X009C)  
MDM\_MCDOpen\_Entry 00260 (0X0104)  
MDM\_MCDOpen\_Exit 00388 (0X0184)  
MDM\_mciClose\_Entry 00262 (0X0106)  
MDM\_mciClose\_Exit 00390 (0X0186)  
MDM\_mciMakeActive\_Entry 00261 (0X0105)  
MDM\_mciMakeActive\_Exit 00389 (0X0185)  
MDM\_mciOpen\_Entry 00259 (0X0103)  
MDM\_mciOpen\_Exit 00387 (0X0183)  
MDM\_mciSendCommandEntry 00258 (0X0102)  
MDM\_mciSendCommandExit 00386 (0X0182)  
MDM\_mciSendStringEntry 00257 (0X0101)  
MDM\_mciSendStringExit 00385 (0X0181)  
MDM\_RESTORE\_Entry 00263 (0X0107)  
MDM\_RESTORE\_EXIT 00391 ( 0X0187)  
MDM\_SAVE\_Entry 00264 (0X0108)  
MDM\_SAVE\_EXIT 00392 (0X0188)  
MDM\_Thread\_call\_Entry 00265 (0X0109)  
MDM\_Thread\_call\_Exit 00393 (0X0189)  
MIO\_DosClose\_Entry 00517 (0X0205)  
MIO\_DosClose\_Exit 00645 (0X0285)  
MIO\_DosDelete\_Entry 00518 (0X0206)  
MIO\_DosDelete\_Exit 00646 (0X0286)  
MIO\_DosOpen\_Entry 00513 (0X0201)  
MIO\_DosOpen\_Exit 00641 (0X0281)  
MIO\_DosRead\_Entry 00514 (0X0202)  
MIO\_DosRead\_Exit 00642 (0X0282)  
MIO\_DosSeek\_Entry 00516 (0X0204)  
MIO\_DosSeek\_Exit 00644 (0X0284)  
MIO\_DosWrite\_Entry 00515 (0X0203)  
MIO\_DosWrite\_Exit 00643 (0X0283)  
MIO\_MemClose\_Entry 00523 (0X020B)  
MIO\_MemClose\_Exit 00651 (0X028B)  
MIO\_MemOpen\_Entry 00519 (0X0207)  
MIO\_MemOpen\_Exit 00647 (0X0287)  
MIO\_MemRead\_Entry 00520 (0X0208)  
MIO\_MemRead\_Exit 00648 (0X0288)  
MIO\_MemSeek\_Entry 00522 (0X020A)  
MIO\_MemSeek\_Exit 00650 (0X028A)

MIO\_MemWrite\_Entry 00521 (0X0209)  
MIO\_MemWrite\_Exit 00649 (0X0289)  
MIO\_MidiClose\_Entry 00557 (0X022D)  
MIO\_MidiClose\_Exit 00685 (0X02AD)  
MIO\_MidiOpen\_Entry 00553 (0X0229)  
MIO\_MidiOpen\_Exit 00681 (0X02A9)  
MIO\_MidiRead\_Entry 00554 (0X022A)  
MIO\_MidiRead\_Exit 00682 (0X02AA)  
MIO\_MidiSeek\_Entry 00556 (0X022C)  
MIO\_MidiSeek\_Exit 00684 (0X02AC)  
MIO\_MidiWrite\_Entry 00555 (0X022B)  
MIO\_MidiWrite\_Exit 00683 (0X02AB)  
MIO\_mmioAdvance\_Entry 00532 (0X0214)  
MIO\_mmioAdvance\_Exit 00660 (0X0294)  
MIO\_mmioAquireSem\_Entry 00535 (0X0217)  
MIO\_mmioAquireSem\_Exit 00663 (0X0297)  
MIO\_mmioAscend\_Entry 00530 (0X0212)  
MIO\_mmioAscend\_Exit 00658 (0X0292)  
MIO\_mmioCFAddElem\_Entry 00550 (0X0226)  
MIO\_mmioCFAddElem\_Exit 00678 (0X02A6)  
MIO\_mmioCFAddEnt\_Entry 00546 (0X0222)  
MIO\_mmioCFAddEnt\_Exit 00674 (0X02A2)  
MIO\_mmioCFAddShrEnt\_Exit 00669 (0X029D)  
MIO\_mmioCFAddShrEnt\_Entry 00541 (0X021D)  
MIO\_mmioCFChgEnt\_Entry 00547 (0X0223)  
MIO\_mmioCFChgEnt\_Exit 00675 (0X02A3)  
MIO\_mmioCFClose\_Entry 00539 (0X021B)  
MIO\_mmioCFClose\_Exit 00667 (0X029B)  
MIO\_mmioCFClsTmpElem\_Exit 00673 (0X02A1)  
MIO\_mmioCFClsTmpElem\_Entry 00545 (0X0221)  
MIO\_mmioCFCopy\_Entry 00548 (0X0224)  
MIO\_mmioCFCopy\_Exit 00676 (0X02A4)  
MIO\_mmioCFDelEnt\_Entry 00549 (0X0225)  
MIO\_mmioCFDelEnt\_Exit 00677 (0X02A5)  
MIO\_mmioCFFndEnt\_Entry 00551 (0X0227)  
MIO\_mmioCFFndEnt\_Exit 00679 (0X02A7)  
MIO\_mmioCFGetInfo\_Entry 00543 (0X021F)  
MIO\_mmioCFGetInfo\_Exit 00671 (0X029F)  
MIO\_mmioCFOpen\_Entry 00538 (0X021A)  
MIO\_mmioCFOpen\_Exit 00666 (0X029A)  
MIO\_mmioCFOpnTmpElem\_Exit 00672 (0X02A0)  
MIO\_mmioCFOpnTmpElem\_Entry 00544 (0X0220)  
MIO\_mmioCFRmvShrEnt\_Entry 00540 (0X021C)  
MIO\_mmioCFRmvShrEnt\_Exit 00668 (0X029C)  
MIO\_mmioCFSetInfo\_Entry 00542 (0X021E)  
MIO\_mmioCFSetInfo\_Exit 00670 (0X029E)  
MIO\_mmioClose\_Entry 00525 (0X020D)  
MIO\_mmioClose\_Exit 00653 (0X028D)  
MIO\_mmioCreateChunk\_Entry 00537 (0X0219)  
MIO\_mmioCreateChunk\_Exit 00665 (0X0299)  
MIO\_mmioDescend\_Entry 00531 (0X0213)  
MIO\_mmioDescend\_Exit 00659 (0X0293)  
MIO\_mmioDiscardSem\_Exit 00664 (0X0298)  
MIO\_mmioDiscardSem\_Entry 00536 (0X0218)  
MIO\_mmioFlush\_Entry 00529 (0X0211)  
MIO\_mmioFlush\_Exit 00657 (0X0291)  
MIO\_mmioldentFile\_Entry 00552 (0X0228)  
MIO\_mmioldentFile\_Exit 00680 (0X02A8)  
MIO\_mmiolnstIOProc\_Exit 00661 (0X0295)  
MIO\_mmiolnstIOProc\_Entry 00533 (0X0215)  
MIO\_mmioOpen\_Entry 00524 (0X020C)  
MIO\_mmioOpen\_Exit 00652 (0X028C)  
MIO\_mmioRead\_Entry 00526 (0X020E)  
MIO\_mmioRead\_Exit 00654 (0X028E)  
MIO\_mmioSeek\_Entry 00528 (0X0210)  
MIO\_mmioSeek\_Exit 00656 (0X0290)  
MIO\_mmioSendMsg\_Entry 00534 (0X0216)  
MIO\_mmioSendMsg\_Exit 00662 (0X0296)  
MIO\_mmioWrite\_Entry 00527 (0X020F)  
MIO\_mmioWrite\_Exit 00655 (0X028F)  
MSH\_RdPlayList\_Entry 00576 (0X0240)  
MSH\_RdPlayList\_Exit 00704 (0X02C0)  
MSH\_WrPlayList\_Entry 00577 (0X0241)



MSH\_WrPlayList\_Exit 00705 (0X02C1)  
MTSH\_GetEmptyEntry 00324 (0X0144)  
MTSH\_GetEmptyExit 00452 (0X01C4)  
MTSH\_ReturnFullEntry 00325 (0X0145)  
MTSH\_ReturnFullExit 00453 (0X01C5)  
RET\_EMPTY\_Entry 00029 (0X001D)  
RET\_EMPTY\_Exit 00157 (0X009D)  
RET\_FULL\_Entry 00030 (0X001E)  
RET\_FULL\_Exit 00158 (0X009E)  
SMH\_ReportEvent0 00032 (0X0020)  
SSM\_AppEvent3\_Entry 00060 (0X003C)  
SSM\_AppEvent3\_Exit 00188 (0X00BC)  
SSM\_AppEventQ\_Entry 00061 (0X003D)  
SSM\_AppEventQ\_Exit 00189 (0X00BD)  
SSM\_CreateBDS 00064 (0X0040)  
SSM\_CreateBDSNotMax 00192 (0X00C0)  
SSM\_IDCDeRegEntry 00025 (0X0019)  
SSM\_IDCDeRegExit 00153 (0X0099)  
SSM\_IDCNotifyEntry 00024 (0X0018)  
SSM\_IDCNotifyExit 00152 (0X0098)  
SSM\_IDCRptEvtEntry 00026 (0X001A)  
SSM\_IDCRptEvtExit 00154 (0X009A)  
SSM\_IDC\_Call\_To\_SHC 00023 (0X0017)  
SSM\_IDC\_Ret\_From\_SHC 00151 (0X0097)  
SSM\_ioctl\_Entry 00022 (0X0016)  
SSM\_ioctl\_Exit 00150 (0X0096)  
SSM\_Negotiate\_error 00160 (0X00A0)  
SSM\_ProcBlock\_Entry 00021 (0X0015)  
SSM\_ProcBlock\_Exit 00149 (0X0095)  
SSM\_ProcRun\_Entry 00020 (0X0014)  
SSM\_ProcRun\_Exit 00148 (0X0094)  
SSM\_QueueEventIOCtl\_Entry 00063 (0X003F)  
SSM\_QueueEventIOCtl\_Exit 00191 (0X00BF)  
SSM\_ShcAssociate\_Entry 00001 (0X0001)  
SSM\_ShcAssociate\_Exit 00129 (0X0081)  
SSM\_ShcClose\_Entry 00002 (0X0002)  
SSM\_ShcClose\_Exit 00130 (0X0082)  
SSM\_ShcCreate\_Entry 00003 (0X0003)  
SSM\_ShcCreate\_Exit 00131 (0X0083)  
SSM\_ShcDestroy\_Entry 00004 (0X0004)  
SSM\_ShcDestroy\_Exit 00132 (0X0084)  
SSM\_ShcDisableEvt\_Entry 00009 (0X0009)  
SSM\_ShcDisableEvt\_Exit 00137 (0X0089)  
SSM\_ShcDisableSync\_Exit 00139 (0X008B)  
SSM\_ShcDisableSync\_Entry 00011 (0X000B)  
SSM\_ShcEnableEvt\_Entry 00008 (0X0008)  
SSM\_ShcEnableEvt\_Exit 00136 (0X0088)  
SSM\_ShcEnableSync\_Entry 00010 (0X000A)  
SSM\_ShcEnableSync\_Exit 00138 (0X008A)  
SSM\_ShcEnumProtocol\_Exit 00143 (0X008F)  
SSM\_ShcEnumProtocol\_Entry 00015 (0X000F)  
SSM\_ShcGetProtocol\_Exit 00141 (0X008D)  
SSM\_ShcGetProtocol\_Entry 00013 (0X000D)  
SSM\_ShcGetTime\_Entry 00012 (0X000C)  
SSM\_ShcGetTime\_Exit 00140 (0X008C)  
SSM\_ShcInstProtocol\_Exit 00142 (0X008E)  
SSM\_ShcInstProtocol\_Entry 00014 (0X000E)  
SSM\_ShcNegotReslt\_Entry 00016 (0X0010)  
SSM\_ShcNegotReslt\_Exit 00144 (0X0090)  
SSM\_ShcRouterRing3\_Entry 00058 (0X003A)  
SSM\_ShcRouterRing3\_Exit 00186 (0X00BA)  
SSM\_ShcRouter\_Entry 00057 (0X0039)  
SSM\_ShcRouter\_Exit 00185 (0X00B9)  
SSM\_ShcSeek\_Entry 00007 (0X0007)  
SSM\_ShcSeek\_Exit 00135 (0X0087)  
SSM\_ShcSendMsg\_Entry 00017 (0X0011)  
SSM\_ShcSendMsg\_Exit 00145 (0X0091)  
SSM\_ShcStart\_Entry 00005 (0X0005)  
SSM\_ShcStart\_Exit 00133 (0X0085)  
SSM\_ShcStop\_Entry 00006 (0X0006)  
SSM\_ShcStop\_Exit 00134 (0X0086)  
SSM\_SmhDeRegister\_Entry 00039 (0X0027)  
SSM\_SmhDeRegister\_Exit 00167 (0X00A7)



SSM\_SmhLockMem\_Entry 00040 (0X0028)  
SSM\_SmhLockMem\_Exit 00168 (0X00A8)  
SSM\_SmhNotifyIOCtrl\_Entry 00062 (0X003E)  
SSM\_SmhNotifyIOCtrl\_Exit 00190 (0X00BE)  
SSM\_SmhNotify\_Entry 00038 (0X0026)  
SSM\_SmhNotify\_Exit 00166 (0X00A6)  
SSM\_SmhRegister\_Entry 00036 (0X0024)  
SSM\_SmhRegister\_Exit 00164 (0X00A4)  
SSM\_SmhReportEvtnt\_Entry 00037 (0X0025)  
SSM\_SmhReportEvtnt\_Exit 00165 (0X00A5)  
SSM\_SmhRing3\_Entry 00035 (0X0023)  
SSM\_SmhRing3\_Exit 00163 (0X00A3)  
SSM\_SmhRptEventSync\_Entry 00059 ( 0X003B)  
SSM\_SpiAddStrConnEntry 00071 (0X0047)  
SSM\_SpiAddStrConnExit 00199 (0X00C7)  
SSM\_SpiAssociate\_Entry 00041 (0X0029)  
SSM\_SpiAssociate\_Exit 00169 (0X00A9)  
SSM\_SpiAssStrNetEntry 00067 (0X0043)  
SSM\_SpiAssStrNetExit 00195 (0X00C3)  
SSM\_SpiCloseStrNetEntry 00068 (0X0044)  
SSM\_SpiCloseStrNetExit 00196 (0X00C4)  
SSM\_SpiConnStrEntry 00069 (0X0045)  
SSM\_SpiConnStrExit 00197 (0X00C5)  
SSM\_SpiCreate\_Entry 00042 (0X002A)  
SSM\_SpiCreate\_Exit 00170 (0X00AA)  
SSM\_SpiDeactStrEntry 00070 (0X0046)  
SSM\_SpiDeactStrExit 00198 (0X00C6)  
SSM\_SpiDestroy\_Entry 00043 (0X002B)  
SSM\_SpiDestroy\_Exit 00171 (0X00AB)  
SSM\_SpiDetSyncMaster\_Entry 00065 (0X0041)  
SSM\_SpiDetSyncMaster\_Exit 00193 (0X00C1)  
SSM\_SpiDisableEvent\_Entry 00054 (0X0036)  
SSM\_SpiDisableEvent\_Exit 00182 (0X00B6)  
SSM\_SpiDisableSync\_Exit 00184 (0X00B8)  
SSM\_SpiDisableSync\_Entry 00056 (0X0038)  
SSM\_SpiEnableEvent\_Entry 00053 (0X0035)  
SSM\_SpiEnableEvent\_Exit 00181 (0X00B5)  
SSM\_SpiEnableSync\_Entry 00055 (0X0037)  
SSM\_SpiEnableSync\_Exit 00183 (0X00B7)  
SSM\_SpiEnumHndlr\_Entry 00045 (0X002D)  
SSM\_SpiEnumHndlr\_Exit 00173 (0X00AD)  
SSM\_SpiEnumProtocol\_Entry 00044 (0X002C)  
SSM\_SpiEnumProtocol\_Exit 00172 (0X00AC)  
SSM\_SpiGetHndlr\_Entry 00046 (0X002E)  
SSM\_SpiGetHndlr\_Exit 00174 (0X00AE)  
SSM\_SpiGetProtocol\_Exit 00175 (0X00AF)  
SSM\_SpiGetProtocol\_Entry 00047 (0X002F)  
SSM\_SpiGetTime\_Entry 00048 (0X0030)  
SSM\_SpiGetTime\_Exit 00176 (0X00B0)  
SSM\_SpiInstProtocol\_Exit 00177 (0X00B1)  
SSM\_SpiInstProtocol\_Entry 00049 (0X0031)  
SSM\_SpiOpenStrNetEntry 00072 (0X0048)  
SSM\_SpiOpenStrNetExit 00200 (0X00C8)  
SSM\_SpiReactStrEntry 00073 (0X0049)  
SSM\_SpiReactStrExit 00201 (0X00C9)  
SSM\_SpiRemStrConnEntry 00074 (0X004A)  
SSM\_SpiRemStrConnExit 00202 (0X00CA)  
SSM\_SpiSeekStrm\_Entry 00050 (0X0032)  
SSM\_SpiSeekStrm\_Exit 00178 (0X00B2)  
SSM\_SpiSeekStrNetEntry 00075 (0X004B)  
SSM\_SpiSeekStrNetExit 00203 (0X00CB)  
SSM\_SpiSendMsg\_Entry 00066 (0X0042)  
SSM\_SpiSendMsg\_Exit 00194 (0X00C2)  
SSM\_SpiStartStrm\_Entry 00051 (0X0033)  
SSM\_SpiStartStrm\_Exit 00179 (0X00B3)  
SSM\_SpiStartStrNetEntry 00076 (0X004C)  
SSM\_SpiStartStrNetExit 00204 (0X00CC)  
SSM\_SpiStopStrm\_Entry 00052 (0X0034)  
SSM\_SpiStopStrm\_Exit 00180 (0X00B4)  
SSM\_SpiStopStrNetEntry 00077 (0X004D)  
SSM\_SpiStopStrNetExit 00205 (0X00CD)  
SSM\_SpiUnassStrNetEntry 00078 (0X004E)  
SSM\_SpiUnassStrNetExit 00206 (0X00CE)

SSM\_xxxUnused1 00187 (0X00BB)  
SVMC\_BitBlitEntry 00618 (0X026A)  
SVMC\_BitBlitExit 00746 (0X02EA)  
SVMC\_BlItUpEntry 00619 (0X026B)  
SVMC\_BlItUpExit 00747 (0X02EB)  
SVMC\_SetBitmapEntry 00617 (0X0269)  
SVMC\_SetBitmapExit 00745 (0X02E9)  
SVMC\_SpiMoveCoorEntry 00620 (0X026C)  
SVMC\_SpiMoveCoorExit 00748 (0X02EC)  
SVSH\_DecompBufferEntry 00612 (0X0264)  
SVSH\_DecompBufferExit 00740 (0X02E4)  
SWVR\_BufferWriteDoneExit 00728 (0X02D8)  
SWVR\_BufferWriteStartEntry 00600 (0X0258)  
SWVR\_CompIndxFrameEntry 00593 (0X0251)  
SWVR\_CompIndxFrameExit 00721 (0X02D1)  
SWVR\_CompRefrFrameEntry 00594 (0X0252)  
SWVR\_CompRefrFrameExit 00722 (0X02D2)  
SWVR\_GetImageEntry 00592 (0X0250)  
SWVR\_GetImageExit 00720 (0X02D0)  
SWVR\_PostBufferEmptyEntry 00596 (0X0254)  
SWVR\_PostBufferEmptyExit 00724 (0X02D4)  
SWVR\_PostBufferFullExit 00723 (0X02D3)  
SWVR\_PostBufferFullEntry 00595 (0X0253)  
SWVR\_PostCountErrorExit 00727 (0X02D7)  
SWVR\_PostCountErrorEntry 00599 (0X0257)  
SWVR\_VCA640CopyEntry 00605 (0X025D)  
SWVR\_VCA640CopyExit 00733 (0X02DD)  
SWVR\_VCABufferCopyEntry 00602 (0X025A)  
SWVR\_VCABufferCopyExit 00730 (0X02DA)  
SWVR\_VCADosDevIOCTLExit 00729 (0X02D9)  
SWVR\_VCADosDevIOCTLEntry 00601 (0X0259)  
SWVR\_VCAtoX2CopyEntry 00604 (0X025C)  
SWVR\_VCAtoX2CopyExit 00732 (0X02DC)  
SWVR\_VCAtoX4CopyEntry 00603 (0X025B)  
SWVR\_VCAtoX4CopyExit 00731 (0X02DB)  
SWVR\_WaitMsgBufferEmptyExit 00726 (0X02D6)  
SWVR\_WaitMsgBufferEmptyEntry 00598 (0X0256)  
SWVR\_WaitMsgBufferFullEntry 00597 (0X0255)  
SWVR\_WaitMsgBufferFullExit 00725 (0X02D5)  
ULDC\_DLLLDroppedEntry 00613 (0X0265)  
ULDC\_DLLLDroppedExit 00741 (0X02E5)  
UNUSED\_HOOK\_109\_018 00018 (0X0012)  
UNUSED\_HOOK\_109\_019 00019 (0X0013)  
UNUSED\_HOOK\_109\_033 00033 (0X0021)  
UNUSED\_HOOK\_109\_034 00034 (0X0022)  
UNUSED\_HOOK\_109\_146 00146 (0X0092)  
UNUSED\_HOOK\_109\_147 00147 (0X0093)  
UNUSED\_HOOK\_109\_161 00161 (0X00A1)  
UNUSED\_HOOK\_109\_162 00162 (0X00A2)

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 1 (0X0001)

### Description

SSM\_ShcAssociate\_Entry

### Tracepoint

Static trace point in Multi-Media Extensions.

### Minor Code

1 (0X0001)

### Trace Groups

SSMSRV

<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RingLvl=%F
-----	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 2 (0X0002)

<u>Description</u>	SSM_ShcClose_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	2 (0X0002)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hid=%F RingLvl=%F
-----	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 3 (0X0003)

<u>Description</u>	SSM_ShcCreate_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	3 (0X0003)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RingLvl=%F DataType=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 4 (0X0004)

<u>Description</u>	SSM_ShcdDestroy_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	4 (0X0004)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RingLvl=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 5 (0X0005)

<u>Description</u>	SSM_ShcdStart_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	5 (0X0005)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RingLvl=%F ulFlags=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 6 (0X0006)

<b><u>Description</u></b>	SSM_ShcStop_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	6 (0X0006)
<b><u>Trace Groups</u></b>	SSMSRV
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hid=%F RingLvl=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 7 (0X0007)

<b><u>Description</u></b>	SSM_ShcSeek_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	7 (0X0007)
<b><u>Trace Groups</u></b>	SSMSRV
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hid=%F RingLvl=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 8 (0X0008)

<b><u>Description</u></b>	SSM_ShcEnableEvt_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	8 (0X0008)
<b><u>Trace Groups</u></b>	

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F RingLvl=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 9 (0X0009)

**Description**

SSM\_ShcdisableEvt\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

9 (0X0009)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RingLvl=%F hevent=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 10 (0X000A)

**Description**

SSM\_ShcdisableSync\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

10 (0X000A)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RingLvl=%F ulFlags=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 11 (0X000B)

<u>Description</u>	SSM_ShcDisableSync_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	11 (0X000B)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RingLvl=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 12 (0X000C)

<u>Description</u>	SSM_ShcGetTime_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	12 (0X000C)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RingLvl=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 13 (0X000D)

<b><u>Description</u></b>	SSM_ShcGetProtocol_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	13 (0X000D)
<b><u>Trace Groups</u></b>	SSMSRV
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hid=%F RingLvl=%F DataType=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 14 (0X000E)

<b><u>Description</u></b>	SSM_ShcInstProtocol_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	14 (0X000E)
<b><u>Trace Groups</u></b>	SSMSRV
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hid=%F RingLvl=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 15 (0X000F)

<b><u>Description</u></b>	SSM_ShcEnumProtocol_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	15 (0X000F)
<b><u>Trace Groups</u></b>	



SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F RingLvl=%F NumKeys=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 16 (0X0010)

**Description**

SSM\_ShcNegotReslt\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

16 (0X0010)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RingLvl=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 17 (0X0011)

**Description**

SSM\_ShcSendMsg\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

17 (0X0011)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RingLvl=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 18  
(0X0012)

<u>Description</u>	UNUSED_HOOK_109_018
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	18 (0X0012)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 19  
(0X0013)

<u>Description</u>	UNUSED_HOOK_109_019
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	19 (0X0013)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 20  
(0X0014)

<b><u>Description</u></b>	SSM_ProcRun_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	20 (0X0014)
<b><u>Trace Groups</u></b>	SSMDD
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	ProcKey=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 21 (0X0015)

<b><u>Description</u></b>	SSM_ProcBlock_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	21 (0X0015)
<b><u>Trace Groups</u></b>	SSMDD
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 22 (0X0016)

<b><u>Description</u></b>	SSM_loctl_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	22 (0X0016)
<b><u>Trace Groups</u></b>	

SSMDD

**Trace Types**

No types assigned.

**Traced Parameters**

Funct/Catg=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 23 (0X0017)

**Description**

SSM\_IDC\_Call\_To\_SHC

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

23 (0X0017)

**Trace Groups**

SSMDD

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F Function=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 24 (0X0018)

**Description**

SSM\_IDCNotifyEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

24 (0X0018)

**Trace Groups**

SSMDD

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 25 (0X0019)

<u>Description</u>	SSM_IDCDeRegEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	25 (0X0019)
<u>Trace Groups</u>	SSMDD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 26 (0X001A)

<u>Description</u>	SSM_IDCRptEvtEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	26 (0X001A)
<u>Trace Groups</u>	SSMDD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

hid=%F ulType=%F ulSubType=%F hstream=%F ulStatus=%F LowParm1=%F HighParm1=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 27 (0X001B)

<b><u>Description</u></b>	GET_EMPTY_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	27 (0X001B)
<b><u>Trace Groups</u></b>	SSMNOTIC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hstreamOwner=%F hbcFull=%F hbcEmpty=%F ulBDSFlag=%F ulSCBFlag=%F ulNumFull=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 28 (0X001C)

<b><u>Description</u></b>	GET_FULL_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	28 (0X001C)
<b><u>Trace Groups</u></b>	SSMNOTIC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hstreamOwner=%F hbcFull=%F hbcEmpty=%F ulBDSFlag=%F ulSCBFlag=%F ulNumFull=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 29 (0X001D)

<b><u>Description</u></b>	RET_EMPTY_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	29 (0X001D)
<b><u>Trace Groups</u></b>	

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F pBuffer=%F pRecord=%F ulLength=%F ulBDSFlag=%F ulSCBFlag=%F ulNumFull=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 30 (0X001E)

**Description**

RET\_FULL\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

30 (0X001E)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F pBuffer=%F pRecord=%F ulLength=%F ulBDSFlag=%F ulSCBFlag=%F ulNumFull=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 31 (0X001F)

**Description**

BUF\_Record\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

31 (0X001F)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

ulNumUsers=%F ulNumRecords=%F rcbHead=%F rcbFull=%F NumAllFull=%F NumSrcOwn=%F NumTgtOwn=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 32 (0X0020)

<u>Description</u>	SMH_ReportEvent0
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	32 (0X0020)
<u>Trace Groups</u>	SSMEVDDC
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F ulType=%F ulSubType=%F ulStatus=%F ulParm1=%F ulParm2=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 33 (0X0021)

<u>Description</u>	UNUSED_HOOK_109_033
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	33 (0X0021)
<u>Trace Groups</u>	SSMEVDDC
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 34 (0X0022)



<b><u>Description</u></b>	UNUSED_HOOK_109_034
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	34 (0X0022)
<b><u>Trace Groups</u></b>	SSMEVDDC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 35 (0X0023)

<b><u>Description</u></b>	SSM_SmhRing3_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	35 (0X0023)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	Function#=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 36 (0X0024)

<b><u>Description</u></b>	SSM_SmhRegister_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	36 (0X0024)
<b><u>Trace Groups</u></b>	

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

Function#=%F ulFlags=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 37 (0X0025)

**Description**

SSM\_SmhReportEvt\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

37 (0X0025)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hevent=%F hid=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 38 (0X0026)

**Description**

SSM\_SmhNotify\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

38 (0X0026)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F Flags=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 39  
(0X0027)

<u>Description</u>	SSM_SmhDeRegister_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	39 (0X0027)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Function#=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 40  
(0X0028)

<u>Description</u>	SSM_SmhLockMem_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	40 (0X0028)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	Function#=%F Flags=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 41  
(0X0029)

<b><u>Description</u></b>	SSM_SpiAssociate_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	41 (0X0029)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hid=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 42 (0X002A)

<b><u>Description</u></b>	SSM_SpiCreate_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	42 (0X002A)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstreamBuf=%F hidSrc=%F hidTgt=%F DataType=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 43 (0X002B)

<b><u>Description</u></b>	SSM_SpiDestroy_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	43 (0X002B)
<b><u>Trace Groups</u></b>	

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 44 (0X002C)

**Description**

SSM\_SpiEnumProtocol\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

44 (0X002C)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 45 (0X002D)

**Description**

SSM\_SpiEnumHndlr\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

45 (0X002D)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 46 (0X002E)

<u>Description</u>	SSM_SpiGetHndlr_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	46 (0X002E)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	N=%F A=%F M=%F E=%F !=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 47 (0X002F)

<u>Description</u>	SSM_SpiGetProtocol_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	47 (0X002F)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hid=%F Datatype=%F DataSubType=%F ullntKey=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 48 (0X0030)

<b><u>Description</u></b>	SSM_SpiGetTime_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	48 (0X0030)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 49 (0X0031)

<b><u>Description</u></b>	SSM_SpiInstProtocol_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	49 (0X0031)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hid=%F Datatype=%F DataSubType=%F ulIntKey=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 50 (0X0032)

<b><u>Description</u></b>	SSM_SpiSeekStrm_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	50 (0X0032)
<b><u>Trace Groups</u></b>	

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F Flags=%F ISeekH=%F ISeekL=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 51 (0X0033)

**Description**

SSM\_SpiStartStrm\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

51 (0X0033)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F Flags=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 52 (0X0034)

**Description**

SSM\_SpiStopStrm\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

52 (0X0034)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F Flags=%F



-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 53 (0X0035)

<u>Description</u>	SSM_SpiEnableEvent_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	53 (0X0035)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F ulType=%F ulSubType=%F Parm1=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 54 (0X0036)

<u>Description</u>	SSM_SpiDisableEvent_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	54 (0X0036)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hevent=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 55 (0X0037)

<b><u>Description</u></b>	SSM_SpiEnableSync_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	55 (0X0037)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstreamMast=%F #slaves=%F mmtimeSync=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 56 (0X0038)

<b><u>Description</u></b>	SSM_SpiDisableSync_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	56 (0X0038)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 57 (0X0039)

<b><u>Description</u></b>	SSM_ShcRouter_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	57 (0X0039)
<b><u>Trace Groups</u></b>	

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 58 (0X003A)

**Description**

SSM\_ShcRouterRing3\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

58 (0X003A)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 59 (0X003B)

**Description**

SSM\_SmhRptEventSync\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

59 (0X003B)

**Trace Groups**

SSMEVDLL

**Trace Types**

No types assigned.

**Traced Parameters**

hstreamSlave=%F mmtimeSlave=%F hstreamMast=%F hidMast=%F mmtimeMast=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 60 (0X003C)

**Description**

SSM\_AppEvent3\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

60 (0X003C)

**Trace Groups**

SSMEVDLL

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hevent=%F ulType=%F ulSubType=%F ulStatus=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 61 (0X003D)

**Description**

SSM\_AppEventQ\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

61 (0X003D)

**Trace Groups**

SSMEVDLL

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hevent=%F ulType=%F ulSubType=%F ulStatus=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 62 (0X003E)

<b><u>Description</u></b>	SSM_SmhNotifyIOCtl_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	62 (0X003E)
<b><u>Trace Groups</u></b>	SSMSRV
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hid=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 63 (0X003F)

<b><u>Description</u></b>	SSM_QueueEventIOCtl_Entry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	63 (0X003F)
<b><u>Trace Groups</u></b>	SSMSRV
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hid=%F ulType=%F ulSubType=%F hstream=%F ulStatus=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 64 (0X0040)

<b><u>Description</u></b>	SSM_CreateBDS
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	64 (0X0040)
<b><u>Trace Groups</u></b>	

SSMBUFF

**Trace Types**

No types assigned.

**Traced Parameters**

ulMinBuf=%F ulMaxBuf=%F ulNumBCB=%F ulHeapSize=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 65 (0X0041)

**Description**

SSM\_SpiDetSyncMaster\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

65 (0X0041)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

#masters=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 66 (0X0042)

**Description**

SSM\_SpiSendMsg\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

66 (0X0042)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F ulMsgType=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 67 (0X0043)

<u>Description</u>	SSM_SpiAssStrNetEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	67 (0X0043)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hnetwork=%F ulFlags=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 68 (0X0044)

<u>Description</u>	SSM_SpiCloseStrNetEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	68 (0X0044)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hnetwork=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 69 (0X0045)

<b><u>Description</u></b>	SSM_SpiConnStrEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	69 (0X0045)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hidSrc=%F hidTgt=%F datatype=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 70 (0X0046)

<b><u>Description</u></b>	SSM_SpiDeactStrEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	70 (0X0046)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 71 (0X0047)

<b><u>Description</u></b>	SSM_SpiAddStrConnEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	71 (0X0047)
<b><u>Trace Groups</u></b>	



SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hnetwork=%F hstream=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 72 (0X0048)

**Description**

SSM\_SpiOpenStrNetEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

72 (0X0048)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 73 (0X0049)

**Description**

SSM\_SpiReactStrEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

73 (0X0049)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 74  
(0X004A)

<u>Description</u>	SSM_SpiRemStrConnEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	74 (0X004A)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hstream=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 75  
(0X004B)

<u>Description</u>	SSM_SpiSeekStrNetEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	75 (0X004B)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hnetwork=%F ulFlags=%F ISeekPoint=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 76  
(0X004C)

<b><u>Description</u></b>	SSM_SpiStartStrNetEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	76 (0X004C)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hnetwork=%F ulFlags=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 77 (0X004D)

<b><u>Description</u></b>	SSM_SpiStopStrNetEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	77 (0X004D)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hnetwork=%F ulFlags=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 78 (0X004E)

<b><u>Description</u></b>	SSM_SpiUnassStrNetEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	78 (0X004E)
<b><u>Trace Groups</u></b>	

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hnetwork=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 80 (0X0050)

**Description**

ACP\_ACPADevIntEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

80 (0X0050)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

Card=%F ID=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 81 (0X0051)

**Description**

ADH\_ADSHIntHandlerEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

81 (0X0051)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F pBuffer=%F ulFlag=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 82 (0X0052)

<u>Description</u>	ADH_SHCCreateEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	82 (0X0052)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hstream=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 83 (0X0053)

<u>Description</u>	ADH_SHCStartEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	83 (0X0053)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hstream=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 84 (0X0054)

<b><u>Description</u></b>	ADH_SHCStopEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	84 (0X0054)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F ulFlags=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 85 (0X0055)

<b><u>Description</u></b>	ADH_GetEmptyEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	85 (0X0055)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hStream=%F RC=%F pBuffer=%F ulFlags=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 86 (0X0056)

<b><u>Description</u></b>	ADH_ReturnFullEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	86 (0X0056)
<b><u>Trace Groups</u></b>	

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F pBuffer=%F ulFlags=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 87 (0X0057)

**Description**

ADH\_GetFullEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

87 (0X0057)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F pBuffer=%F ulFlags=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 88 (0X0058)

**Description**

ADH\_ReturnEmptyEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

88 (0X0058)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F pBuffer=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 89 (0X0059)

<u>Description</u>	ADH_ReportEOSEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	89 (0X0059)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hStream=%F RC=%F pBuffer=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 90 (0X005A)

<u>Description</u>	ADH_ReportErrorEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	90 (0X005A)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 91 (0X005B)

Description



ADH\_ReportPLCueEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

91 (0X005B)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 92 (0X005C)

**Description**

ADH\_ReportCueTimeEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

92 (0X005C)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 93 (0X005D)

**Description**

ADH\_ReportCueDataEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

93 (0X005D)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**  
No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 94 (0X005E)

**Description** ADH\_ReportDataUnderEntry

**Tracepoint** Static trace point in Multi-Media Extensions.

**Minor Code** 94 (0X005E)

**Trace Groups** AD SH

**Trace Types** No types assigned.

**Traced Parameters**

hStream=%F ulStatus=%F ulStateFlg=%F CurrBufIndex=%F EOSBufIndex=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 95 (0X005F)

**Description** ADH\_ReportSyncOverEntry

**Tracepoint** Static trace point in Multi-Media Extensions.

**Minor Code** 95 (0X005F)

**Trace Groups** AD SH

**Trace Types** No types assigned.

**Traced Parameters**

hStream=%F ulStatus=%F ulStateFlg=%F CurrBufIndex=%F EOSBufIndex=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 96

## (0X0060)

<u>Description</u>	ADH_EventHandlerEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	96 (0X0060)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hStream=%F ulStreamTime=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 97 (0X0061)

<u>Description</u>	ACP_DevReportError
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	97 (0X0061)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Error(DSP/MME)=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 101 (0X0065)

<u>Description</u>	ADSH_AddBufEntry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

101 (0X0065)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 102 (0X0066)

**Description**

ADSH\_RemoveBufEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

102 (0X0066)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 103 (0X0067)

**Description**

ADSH\_IDCWriteEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

103 (0X0067)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 104 (0X0068)

<u>Description</u>	ADSH_GetBufferEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	104 (0X0068)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code: 105 (0X0069)

<u>Description</u>	ADSH_RetBufferEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	105 (0X0069)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code:

## 106 (0X006A)

<u>Description</u>	ADSH_DLLIntEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	106 (0X006A)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 107 (0X006B)

<u>Description</u>	ADSH_IDCCControlEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	107 (0X006B)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 108 (0X006C)

<u>Description</u>	ADSH_IDCSetupEntry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

108 (0X006C)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 109 (0X006D)

**Description**

ADSH\_SendBufferEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

109 (0X006D)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 112 (0X0070)

**Description**

CDSH\_StartCdReadIoctl

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

112 (0X0070)

**Trace Groups**

CD

**Trace Types**

No types assigned.

Traced Parameters

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 113 (0X0071)

<u>Description</u>	CDSH_StartCddaRead
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	113 (0X0071)
<u>Trace Groups</u>	CD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 114 (0X0072)

<u>Description</u>	CDSH_StartCddaWrite
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	114 (0X0072)
<u>Trace Groups</u>	CD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:



## 129 (0X0081)

**Description**

SSM\_ShcAssociate\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

129 (0X0081)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 130 (0X0082)

**Description**

SSM\_ShcClose\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

130 (0X0082)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 131 (0X0083)

**Description**

SSM\_ShcCreate\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

131 (0X0083)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 132 (0X0084)

**Description**

SSM\_ShcdDestroy\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

132 (0X0084)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 133 (0X0085)

**Description**

SSM\_ShcdStart\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

133 (0X0085)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 134 (0X0086)

**Description**

SSM\_ShcStop\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

134 (0X0086)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 135 (0X0087)

**Description**

SSM\_ShcSeek\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

135 (0X0087)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

## 136 (0X0088)

<u>Description</u>	SSM_ShcEnableEvt_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	136 (0X0088)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 137 (0X0089)

<u>Description</u>	SSM_ShcDisableEvt_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	137 (0X0089)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 138 (0X008A)

<u>Description</u>	SSM_ShcEnableSync_Exit
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

138 (0X008A)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 139 (0X008B)

**Description**

SSM\_ShcdisableSync\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

139 (0X008B)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 140 (0X008C)

**Description**

SSM\_ShcdGetTime\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

140 (0X008C)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 141 (0X008D)

**Description**

SSM\_ShcGetProtocol\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

141 (0X008D)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 142 (0X008E)

**Description**

SSM\_ShcInstProtocol\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

142 (0X008E)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

## 143 (0X008F)

<u>Description</u>	SSM_ShcEnumProtocol_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	143 (0X008F)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 144 (0X0090)

<u>Description</u>	SSM_ShcNegotReslt_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	144 (0X0090)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 145 (0X0091)

<u>Description</u>	SSM_ShcSendMsg_Exit
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

145 (0X0091)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 146 (0X0092)

**Description**

UNUSED\_HOOK\_109\_146

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

146 (0X0092)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 147 (0X0093)

**Description**

UNUSED\_HOOK\_109\_147

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

147 (0X0093)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.



Traced Parameters

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 148 (0X0094)

<u>Description</u>	SSM_ProcRun_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	148 (0X0094)
<u>Trace Groups</u>	SSMDD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 149 (0X0095)

<u>Description</u>	SSM_ProcBlock_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	149 (0X0095)
<u>Trace Groups</u>	SSMDD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

UnblockFlag=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

## 150 (0X0096)

<u>Description</u>	SSM_Iocctl_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	150 (0X0096)
<u>Trace Groups</u>	SSMDD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Func/Catg=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 151 (0X0097)

<u>Description</u>	SSM_IDC_Ret_From_SHC
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	151 (0X0097)
<u>Trace Groups</u>	SSMDD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hid=%F Function=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 152 (0X0098)

<u>Description</u>	SSM_IDCNotifyExit
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

152 (0X0098)

**Trace Groups**

SSMDD

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 153 (0X0099)

**Description**

SSM\_IDCDeRegExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

153 (0X0099)

**Trace Groups**

SSMDD

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 154 (0X009A)

**Description**

SSM\_IDCRptEvtExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

154 (0X009A)

**Trace Groups**

SSMDD

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F RC=%F LowParm2=%F HighParm2=%F LowParm3=%F HighParm3=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 155 (0X009B)

**Description**

GET\_EMPTY\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

155 (0X009B)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F pBufferP=%F pBuffer=%F pRecord=%F ulLength=%F hbcdbFull=%F hbcdbEmpty=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 156 (0X009C)

**Description**

GET\_FULL\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

156 (0X009C)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F pBufferP=%F pBuffer=%F pRecord=%F ulLength=%F hbcdbFull=%F hbcdbEmpty=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

## 157 (0X009D)

**Description**

RET\_EMPTY\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

157 (0X009D)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hstreamOwner=%F pBufferP=%F hbcBThis=%F hbcBFull=%F hbcBEmpty=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 158 (0X009E)

**Description**

RET\_FULL\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

158 (0X009E)

**Trace Groups**

SSMNOTIC

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hstreamOwner=%F pBufferP=%F hbcBThis=%F hbcBFull=%F hbcBEmpty=%F rc=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 159 (0X009F)

**Description**

BUF\_Record\_Exit

**Tracepoint**

	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	159 (0X009F)
<b><u>Trace Groups</u></b>	SSMNOTIC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	ulNumUsers=%F ulNumRecords=%F rcbHead=%F rcbFull=%F NumAllFull=%F NumSrcOwn=%F NumATgtOwn=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 160 (0X00A0)

<b><u>Description</u></b>	SSM_Negotiate_error
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	160 (0X00A0)
<b><u>Trace Groups</u></b>	SSMEVDDC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F ErrorStatus=%F KeyDataType=%F KeySubType=%F KeyIntKey=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 161 (0X00A1)

<b><u>Description</u></b>	UNUSED_HOOK_109_161
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	161 (0X00A1)
<b><u>Trace Groups</u></b>	SSMEVDDC
<b><u>Trace Types</u></b>	

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 162 (0X00A2)

**Description**

UNUSED\_HOOK\_109\_162

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

162 (0X00A2)

**Trace Groups**

SSMEVDDC

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 163 (0X00A3)

**Description**

SSM\_SmhRing3\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

163 (0X00A3)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

Function#=%F RC=%F

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 164 (0X00A4)

<u>Description</u>	SSM_SmhRegister_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	164 (0X00A4)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hidSrc=%F hidTgt=%F

# Multi-Media Extensions Major Code: 0X006D Minor Code: 165 (0X00A5)

<u>Description</u>	SSM_SmhReportEvt_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	165 (0X00A5)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hevent=%F RC=%F

# Multi-Media Extensions Major Code: 0X006D Minor Code: 166 (0X00A6)

<u>Description</u>
--------------------



SSM\_SmhNotify\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

166 (0X00A6)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 167 (0X00A7)

**Description**

SSM\_SmhDeRegister\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

167 (0X00A7)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

Function#=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 168 (0X00A8)

**Description**

SSM\_SmhLockMem\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

168 (0X00A8)

**Trace Groups**

SSMAPI

<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Function#=%F Flags=%F RC=%F
	-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 169 (0X00A9)

<u>Description</u>	SSM_SpiAssociate_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	169 (0X00A9)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstream=%F hid=%F RC=%F
	-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 170 (0X00AA)

<u>Description</u>	SSM_SpiCreate_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	170 (0X00AA)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hstreamBuf=%F hstream=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 171 (0X00AB)

<u>Description</u>	SSM_SpiDestroy_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	171 (0X00AB)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hstream=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 172 (0X00AC)

<u>Description</u>	SSM_SpiEnumProtocol_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	172 (0X00AC)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 173 (0X00AD)

<b><u>Description</u></b>	SSM_SpiEnumHndlr_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	173 (0X00AD)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	#Handlers=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 174 (0X00AE)

<b><u>Description</u></b>	SSM_SpiGetHndlr_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	174 (0X00AE)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hidSrc=%F hidTgt=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 175 (0X00AF)

<b><u>Description</u></b>	SSM_SpiGetProtocol_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	175 (0X00AF)
<b><u>Trace Groups</u></b>	

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F Datatype=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 176 (0X00B0)

**Description**

SSM\_SpiGetTime\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

176 (0X00B0)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F mmtimeH=%F mmtimeL=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 177 (0X00B1)

**Description**

SSM\_SpiInstProtocol\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

177 (0X00B1)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F Datatype=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 178 (0X00B2)

<u>Description</u>	SSM_SpiSeekStrm_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	178 (0X00B2)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hstream=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 179 (0X00B3)

<u>Description</u>	SSM_SpiStartStrm_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	179 (0X00B3)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hstream=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 180 (0X00B4)

<b><u>Description</u></b>	SSM_SpiStopStrm_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	180 (0X00B4)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F RC=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 181 (0X00B5)

<b><u>Description</u></b>	SSM_SpiEnableEvent_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	181 (0X00B5)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hevent=%F RC=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 182 (0X00B6)

<b><u>Description</u></b>	SSM_SpiDisableEvent_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	182 (0X00B6)
<b><u>Trace Groups</u></b>	

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hevent=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 183 (0X00B7)

**Description**

SSM\_SpiEnableSync\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

183 (0X00B7)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstreamMast=%F #slaves=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 184 (0X00B8)

**Description**

SSM\_SpiDisableSync\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

184 (0X00B8)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F RC=%F



-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 185 (0X00B9)

<u>Description</u>	SSM_ShcRouter_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	185 (0X00B9)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 186 (0X00BA)

<u>Description</u>	SSM_ShcRouterRing3_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	186 (0X00BA)
<u>Trace Groups</u>	SSMSRV
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 187 (0X00BB)

<b><u>Description</u></b>	SSM_xxxUnused1
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	187 (0X00BB)
<b><u>Trace Groups</u></b>	SSMEVDLL
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 188 (0X00BC)

<b><u>Description</u></b>	SSM_AppEvent3_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	188 (0X00BC)
<b><u>Trace Groups</u></b>	SSMEVDLL
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hevent=%F hid=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 189 (0X00BD)

<b><u>Description</u></b>	SSM_AppEventQ_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	189 (0X00BD)
<b><u>Trace Groups</u></b>	

SSMEVDLL

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hevent=%F hid=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 190 (0X00BE)

**Description**

SSM\_SmhNotifyIOCtl\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

190 (0X00BE)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F hid0/RC3=%F Flags=%F ulGetNumEn=%F ulRetNumEn=%F RC0=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 191 (0X00BF)

**Description**

SSM\_QueueEventIOCtl\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

191 (0X00BF)

**Trace Groups**

SSMSRV

**Trace Types**

No types assigned.

**Traced Parameters**

hid=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 192 (0X00C0)

<u>Description</u>	SSM_CreateBDSNotMax
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	192 (0X00C0)
<u>Trace Groups</u>	SSMBUFF
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	ulMinBuf=%F ulMaxBuf=%F ulNumBCB=%F ulHeapSize=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 193 (0X00C1)

<u>Description</u>	SSM_SpiDetSyncMaster_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	193 (0X00C1)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	#masters=%F RC=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 194 (0X00C2)

<b><u>Description</u></b>	SSM_SpiSendMsg_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	194 (0X00C2)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F hid=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 195 (0X00C3)

<b><u>Description</u></b>	SSM_SpiAssStrNetExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	195 (0X00C3)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hnetwork=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 196 (0X00C4)

<b><u>Description</u></b>	SSM_SpiCloseStrNetExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	196 (0X00C4)
<b><u>Trace Groups</u></b>	

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 197 (0X00C5)

**Description**

SSM\_SpiConnStrExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

197 (0X00C5)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hidSrc=%F hidTgt=%F hstream=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 198 (0X00C6)

**Description**

SSM\_SpiDeactStrExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

198 (0X00C6)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 199 (0X00C7)

<u>Description</u>	SSM_SpiAddStrConnExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	199 (0X00C7)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hnetwork=%F hstream=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 200 (0X00C8)

<u>Description</u>	SSM_SpiOpenStrNetExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	200 (0X00C8)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hnetwork=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 201 (0X00C9)

<b><u>Description</u></b>	SSM_SpiReactStrExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	201 (0X00C9)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F RC=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 202 (0X00CA)

<b><u>Description</u></b>	SSM_SpiRemStrConnExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	202 (0X00CA)
<b><u>Trace Groups</u></b>	SSMAPI
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hstream=%F RC=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 203 (0X00CB)

<b><u>Description</u></b>	SSM_SpiSeekStrNetExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	203 (0X00CB)
<b><u>Trace Groups</u></b>	



SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 204 (0X00CC)

**Description**

SSM\_SpiStartStrNetExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

204 (0X00CC)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 205 (0X00CD)

**Description**

SSM\_SpiStopStrNetExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

205 (0X00CD)

**Trace Groups**

SSMAPI

**Trace Types**

No types assigned.

**Traced Parameters**

hnetwork=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 206 (0X00CE)

<u>Description</u>	SSM_SpiUnassStrNetExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	206 (0X00CE)
<u>Trace Groups</u>	SSMAPI
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hnetwork=%F RC=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 208 (0X00D0)

<u>Description</u>	ACP_ACPADevIntExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	208 (0X00D0)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Card=%F ID=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 209 (0X00D1)

<b><u>Description</u></b>	ADH_ADShIntHandlerExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	209 (0X00D1)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hStream=%F RC=%F ulStateFlg=%F CurrBufIndex=%F EOSBufIndex=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 210 (0X00D2)

<b><u>Description</u></b>	ADH_SHCCreateExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	210 (0X00D2)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hStream=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 211 (0X00D3)

<b><u>Description</u></b>	ADH_SHCStartExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	211 (0X00D3)
<b><u>Trace Groups</u></b>	

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F ulStateFlg=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 212 (0X00D4)

**Description**

ADH\_SHCStopExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

212 (0X00D4)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F ulStateFlg=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 213 (0X00D5)

**Description**

ADH\_GetEmptyExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

213 (0X00D5)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

hStream=%F RC=%F pBuffer=%F ulFlags=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 214 (0X00D6)

<u>Description</u>	ADH_ReturnFullExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	214 (0X00D6)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hStream=%F RC=%F pBuffer=%F ulFlags=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 215 (0X00D7)

<u>Description</u>	ADH_GetFullExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	215 (0X00D7)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hStream=%F RC=%F pBuffer=%F ulFlags=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 216 (0X00D8)

<b><u>Description</u></b>	ADH_ReturnEmptyExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	216 (0X00D8)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hStream=%F RC=%F pBuffer=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 217 (0X00D9)

<b><u>Description</u></b>	ADH_ReportEOSExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	217 (0X00D9)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hStream=%F RC=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 218 (0X00DA)

<b><u>Description</u></b>	ADH_ReportErrorExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	218 (0X00DA)
<b><u>Trace Groups</u></b>	

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 219 (0X00DB)

**Description**

ADH\_ReportPLCueExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

219 (0X00DB)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 220 (0X00DC)

**Description**

ADH\_ReportCueTimeExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

220 (0X00DC)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code:

# 221 (0X00DD)

<u>Description</u>	ADH_ReportCueDataExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	221 (0X00DD)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 222 (0X00DE)

<u>Description</u>	ADH_ReportDataUnderExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	222 (0X00DE)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	hStream=%F ulStatus=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 223 (0X00DF)

<u>Description</u>	ADH_ReportSyncOverExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.



<b><u>Minor Code</u></b>	223 (0X00DF)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

hStream=%F ulStatus=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 224 (0X00E0)

<b><u>Description</u></b>	ADH_EventHandlerExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	224 (0X00E0)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

hStream=%F ulStreamTime=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 225 (0X00E1)

<b><u>Description</u></b>	ACP_DevReportError
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	225 (0X00E1)
<b><u>Trace Groups</u></b>	ADSH
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

Error(DSP/MME)=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 229 (0X00E5)

**Description**

ADSH\_AddBufExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

229 (0X00E5)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 230 (0X00E6)

**Description**

ADSH\_RemoveBufExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

230 (0X00E6)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

---

## Multi-Media Extensions Major Code: 0X006D Minor Code:

# 231 (0X00E7)

<u>Description</u>	ADSH_IDCWriteExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	231 (0X00E7)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 232 (0X00E8)

<u>Description</u>	ADSH_GetBufferExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	232 (0X00E8)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 233 (0X00E9)

<u>Description</u>	ADSH_RetBufferExit
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

233 (0X00E9)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 234 (0X00EA)

**Description**

ADSH\_DLLIntExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

234 (0X00EA)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 235 (0X00EB)

**Description**

ADSH\_IDCCControlExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

235 (0X00EB)

**Trace Groups**

ADSH

**Trace Types**

No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 236 (0X00EC)

<u>Description</u>	ADSH_IDCSetupExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	236 (0X00EC)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code: 237 (0X00ED)

<u>Description</u>	ADSH_SendBufferExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	237 (0X00ED)
<u>Trace Groups</u>	ADSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 240 (0X00F0)

<u>Description</u>	CDSH_StopCdReadIoctl
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	240 (0X00F0)
<u>Trace Groups</u>	CD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 241 (0X00F1)

<u>Description</u>	CDSH_StopCddaRead
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	241 (0X00F1)
<u>Trace Groups</u>	CD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 242 (0X00F2)

<u>Description</u>	CDSH_StopCddaWrite
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

242 (0X00F2)

**Trace Groups**

CD

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 257 (0X0101)

**Description**

MDM\_mciSendStringEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

257 (0X0101)

**Trace Groups**

MMSNDSTR

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 258 (0X0102)

**Description**

MDM\_mciSendCommandEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

258 (0X0102)

**Trace Groups**

MMSNDCMD

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Message=%F UserParm=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 259 (0X0103)

**Description**

MDM\_mciOpen\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

259 (0X0103)

**Trace Groups**

MCIOPEN

**Trace Types**

No types assigned.

**Traced Parameters**

UserParm=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 260 (0X0104)

**Description**

MDM\_MCDOpen\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

260 (0X0104)

**Trace Groups**

MCIOPEN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Type=%F UserParm=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:



261 (0X0105)

<u>Description</u>	MDM_mciMakeActive_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	261 (0X0105)
<u>Trace Groups</u>	MCIOPEN
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	ID=%F Type=%F RU=%F Class=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 262 (0X0106)

<u>Description</u>	MDM_mciClose_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	262 (0X0106)
<u>Trace Groups</u>	MCICLOSE
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	ID=%F UserParm=%F

Multi-Media Extensions Major Code: 0X006D Minor Code: 263 (0X0107)

<u>Description</u>	MDM_RESTORE_Entry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

263 (0X0107)

**Trace Groups**

MMRESMAN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Type=%F RU=%F Class=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 264 (0X0108)

**Description**

MDM\_SAVE\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

264 (0X0108)

**Trace Groups**

MMRESMAN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Type=%F RU=%F Class=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 265 (0X0109)

**Description**

MDM\_Thread\_call\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

265 (0X0109)

**Trace Groups**

MMRESMAN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Message=%F UserParm=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 324 (0X0144)

**Description**

MTSH\_GetEmptyEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

324 (0X0144)

**Trace Groups**

MTSHREAD

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 325 (0X0145)

**Description**

MTSH\_ReturnFullEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

325 (0X0145)

**Trace Groups**

MTSHREAD

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 385 (0X0181)

<u>Description</u>	MDM_mciSendStringExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	385 (0X0181)
<u>Trace Groups</u>	MMSNDSTR
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	ID=%F RC=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 386 (0X0182)

<u>Description</u>	MDM_mciSendCommandExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	386 (0X0182)
<u>Trace Groups</u>	MMSNDCMD
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	ID=%F RC=%F UserParm=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 387 (0X0183)

<u>Description</u>	MDM_mciOpen_Exit
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

387 (0X0183)

**Trace Groups**

MCIOPEN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F RC=%F UserParm=%F rc=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 388 (0X0184)

**Description**

MDM\_MCDOpen\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

388 (0X0184)

**Trace Groups**

MCIOPEN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Type=%F RU=%F Class=%F UserParm=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 389 (0X0185)

**Description**

MDM\_mciMakeActive\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

389 (0X0185)

**Trace Groups**

MCIOPEN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Type=%F RU=%F Class=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 390 (0X0186)

**Description**

MDM\_mciClose\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

390 (0X0186)

**Trace Groups**

MCICLOSE

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F RC=%F UserParm=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 391 (0X0187)

**Description**

MDM\_RESTORE\_EXIT

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

391 (0X0187)

**Trace Groups**

MMRESMAN

**Trace Types**

No types assigned.

**Traced Parameters**

ID=%F Type=%F RU=%F Class=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 392 (0X0188)

<u>Description</u>	MDM_SAVE_EXIT
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	392 (0X0188)
<u>Trace Groups</u>	MMRESMAN
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	ID=%F Type=%F RU=%F Class=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 393 (0X0189)

<u>Description</u>	MDM_Thread_call_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	393 (0X0189)
<u>Trace Groups</u>	MMRESMAN
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	ID=%F Message=%F UserParm=%F RC=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 452 (0X01C4)

<u>Description</u>	MTSH_GetEmptyExit
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

452 (0X01C4)

**Trace Groups**

MTSHREAD

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F RC=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 453 (0X01C5)

**Description**

MTSH\_ReturnFullExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

453 (0X01C5)

**Trace Groups**

MTSHREAD

**Trace Types**

No types assigned.

**Traced Parameters**

hstream=%F RC=%F ulFlags=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 513 (0X0201)

**Description**

MIO\_DosOpen\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

513 (0X0201)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.



Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 514 (0X0202)

<u>Description</u>	MIO_DosRead_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	514 (0X0202)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code: 515 (0X0203)

<u>Description</u>	MIO_DosWrite_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	515 (0X0203)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code:

516 (0X0204)

<u>Description</u>	MIO_DosSeek_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	516 (0X0204)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:  
517 (0X0205)

<u>Description</u>	MIO_DosClose_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	517 (0X0205)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:  
518 (0X0206)

<u>Description</u>	MIO_DosDelete_Entry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

518 (0X0206)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 519 (0X0207)

**Description**

MIO\_MemOpen\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

519 (0X0207)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 520 (0X0208)

**Description**

MIO\_MemRead\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

520 (0X0208)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

Traced Parameters

Multi-Media Extensions Major Code: 0X006D Minor Code: 521 (0X0209)

<u>Description</u>	MIO_MemWrite_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	521 (0X0209)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code: 522 (0X020A)

<u>Description</u>	MIO_MemSeek_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	522 (0X020A)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 523 (0X020B)

<u>Description</u>	MIO_MemClose_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	523 (0X020B)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 524 (0X020C)

<u>Description</u>	MIO_mmioOpen_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	524 (0X020C)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 525 (0X020D)

<u>Description</u>	MIO_mmioClose_Entry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

525 (0X020D)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 526 (0X020E)

**Description**

MIO\_mmioRead\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

526 (0X020E)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

cch=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 527 (0X020F)

**Description**

MIO\_mmioWrite\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

527 (0X020F)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

cch=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 528 (0X0210)

**Description**

MIO\_mmioSeek\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

528 (0X0210)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

IOffset=%F IOrgin=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 529 (0X0211)

**Description**

MIO\_mmioFlush\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

529 (0X0211)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 530 (0X0212)

<u>Description</u>	MIO_mmioAscend_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	530 (0X0212)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 531 (0X0213)

<u>Description</u>	MIO_mmioDescend_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	531 (0X0213)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 532 (0X0214)

<u>Description</u>	MIO_mmioAdvance_Entry
<u>Tracepoint</u>	



Static trace point in Multi-Media Extensions.

**Minor Code**

532 (0X0214)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 533 (0X0215)

**Description**

MIO\_mmioInstIOProc\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

533 (0X0215)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 534 (0X0216)

**Description**

MIO\_mmioSendMsg\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

534 (0X0216)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

Traced Parameters

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 535 (0X0217)

<u>Description</u>	MIO_mmioAquireSem_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	535 (0X0217)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 536 (0X0218)

<u>Description</u>	MIO_mmioDiscardSem_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	536 (0X0218)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 537 (0X0219)

<u>Description</u>	MIO_mmioCreateChunk_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	537 (0X0219)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 538 (0X021A)

<u>Description</u>	MIO_mmioCFOpen_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	538 (0X021A)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 539 (0X021B)

<u>Description</u>	MIO_mmioCFClose_Entry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

539 (0X021B)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 540 (0X021C)

**Description**

MIO\_mmioCFRmvShrEnt\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

540 (0X021C)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 541 (0X021D)

**Description**

MIO\_mmioCFAddShrEnt\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

541 (0X021D)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

Traced Parameters

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 542 (0X021E)

<u>Description</u>	MIO_mmioCFSetInfo_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	542 (0X021E)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 543 (0X021F)

<u>Description</u>	MIO_mmioCFGetInfo_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	543 (0X021F)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 544 (0X0220)

<u>Description</u>	MIO_mmioCFOpnTmpElem_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	544 (0X0220)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 545 (0X0221)

<u>Description</u>	MIO_mmioCFClsTmpElem_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	545 (0X0221)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 546 (0X0222)

<u>Description</u>	MIO_mmioCFAddEnt_Entry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

546 (0X0222)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 547 (0X0223)

**Description**

MIO\_mmioCFChgEnt\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

547 (0X0223)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 548 (0X0224)

**Description**

MIO\_mmioCFCopy\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

548 (0X0224)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

Traced Parameters

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 549 (0X0225)

<u>Description</u>	MIO_mmioCFDelEnt_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	549 (0X0225)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 550 (0X0226)

<u>Description</u>	MIO_mmioCFAddElem_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	550 (0X0226)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:



# 551 (0X0227)

<u>Description</u>	MIO_mmioCFFndEnt_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	551 (0X0227)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 552 (0X0228)

<u>Description</u>	MIO_mmioldentFile_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	552 (0X0228)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 553 (0X0229)

<u>Description</u>	MIO_MidiOpen_Entry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

553 (0X0229)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 554 (0X022A)

**Description**

MIO\_MidiRead\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

554 (0X022A)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 555 (0X022B)

**Description**

MIO\_MidiWrite\_Entry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

555 (0X022B)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

Traced Parameters

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 556 (0X022C)

<u>Description</u>	MIO_MidiSeek_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	556 (0X022C)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 557 (0X022D)

<u>Description</u>	MIO_MidiClose_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	557 (0X022D)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code:

# 576 (0X0240)

<u>Description</u>	MSH_RdPlayList_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	576 (0X0240)
<u>Trace Groups</u>	MSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Operation=%F Operand1=%F Operand2=%F Operand3=%F Entry#=%F

# Multi-Media Extensions Major Code: 0X006D Minor Code: 577 (0X0241)

<u>Description</u>	MSH_WrPlayList_Entry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	577 (0X0241)
<u>Trace Groups</u>	MSH
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Operation=%F Operand1=%F Operand2=%F Operand3=%F Entry#=%F

# Multi-Media Extensions Major Code: 0X006D Minor Code: 592 (0X0250)

<u>Description</u>	SWVR_GetImageEntry
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

592 (0X0250)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 593 (0X0251)

**Description**

SWVR\_CompIdxFrameEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

593 (0X0251)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 594 (0X0252)

**Description**

SWVR\_CompRefrFrameEntry

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

594 (0X0252)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 595 (0X0253)

<u>Description</u>	SWVR_PostBufferFullEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	595 (0X0253)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 596 (0X0254)

<u>Description</u>	SWVR_PostBufferEmptyEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	596 (0X0254)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 597 (0X0255)

<u>Description</u>	SWVR_WaitMsgBufferFullEntry
--------------------	-----------------------------

<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	597 (0X0255)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 598 (0X0256)

<b><u>Description</u></b>	SWVR_WaitMsgBufferEmptyEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	598 (0X0256)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 599 (0X0257)

<b><u>Description</u></b>	SWVR_PostCountErrorEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	599 (0X0257)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

PostCount=%d=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 600 (0X0258)

<b><u>Description</u></b>	SWVR_BufferWriteStartEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	600 (0X0258)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	ByteCount=%d=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 601 (0X0259)

<b><u>Description</u></b>	SWVR_VCADosDevIOCTLEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	601 (0X0259)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 602 (0X025A)



<b><u>Description</u></b>	SWVR_VCABufferCopyEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	602 (0X025A)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

## Multi-Media Extensions Major Code: 0X006D Minor Code: 603 (0X025B)

<b><u>Description</u></b>	SWVR_VCAtoX4CopyEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	603 (0X025B)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

## Multi-Media Extensions Major Code: 0X006D Minor Code: 604 (0X025C)

<b><u>Description</u></b>	SWVR_VCAtoX2CopyEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	604 (0X025C)
<b><u>Trace Groups</u></b>	MMIO

**Trace Types** No types assigned.

**Traced Parameters** No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 605 (0X025D)

**Description** SWVR\_VCA640CopyEntry

**Tracepoint** Static trace point in Multi-Media Extensions.

**Minor Code** 605 (0X025D)

**Trace Groups** MMIO

**Trace Types** No types assigned.

**Traced Parameters** No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 612 (0X0264)

**Description** SVSH\_DecompBufferEntry

**Tracepoint** Static trace point in Multi-Media Extensions.

**Minor Code** 612 (0X0264)

**Trace Groups** MMIO

**Trace Types** No types assigned.

**Traced Parameters**

hStream=%F Frame#=%F mmtSlave=%F mmtMaster=%F FramIntvl=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code:

# 613 (0X0265)

<u>Description</u>	ULDC_DLLDroppedEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	613 (0X0265)
<u>Trace Groups</u>	SVSHSYNC
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	Frame#=%F Tolerance=%F mmtNxtFrame=%F mmtCurrent=%F SyncMethod=%F

## Multi-Media Extensions Major Code: 0X006D Minor Code: 617 (0X0269)

<u>Description</u>	SVMC_SetBitmapEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	617 (0X0269)
<u>Trace Groups</u>	SVSHSYNC
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

## Multi-Media Extensions Major Code: 0X006D Minor Code: 618 (0X026A)

<u>Description</u>	SVMC_BitBlitEntry
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.

<b><u>Minor Code</u></b>	618 (0X026A)
<b><u>Trace Groups</u></b>	SVSHSYNC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 619 (0X026B)

<b><u>Description</u></b>	SVMC_BlItUpEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	619 (0X026B)
<b><u>Trace Groups</u></b>	SVSHSYNC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 620 (0X026C)

<b><u>Description</u></b>	SVMC_SpiMoveCoorEntry
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	620 (0X026C)
<b><u>Trace Groups</u></b>	SVSHSYNC
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

# Multi-Media Extensions Major Code: 0X006D Minor Code: 621 (0X026D)

<u>Description</u>	
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	621 (0X026D)
<u>Trace Groups</u>	SVSHSYNC
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

---

# Multi-Media Extensions Major Code: 0X006D Minor Code: 641 (0X0281)

<u>Description</u>	MIO_DosOpen_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	641 (0X0281)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

# Multi-Media Extensions Major Code: 0X006D Minor Code: 642 (0X0282)

<u>Description</u>	MIO_DosRead_Exit
--------------------	------------------

<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	642 (0X0282)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 643 (0X0283)

<b><u>Description</u></b>	MIO_DosWrite_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	643 (0X0283)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 644 (0X0284)

<b><u>Description</u></b>	MIO_DosSeek_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	644 (0X0284)
<b><u>Trace Groups</u></b>	MMIO

**Trace Types**                      No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 645 (0X0285)

**Description**                      MIO\_DosClose\_Exit

**Tracepoint**                      Static trace point in Multi-Media Extensions.

**Minor Code**                      645 (0X0285)

**Trace Groups**                      MMIO

**Trace Types**                      No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 646 (0X0286)

**Description**                      MIO\_DosDelete\_Exit

**Tracepoint**                      Static trace point in Multi-Media Extensions.

**Minor Code**                      646 (0X0286)

**Trace Groups**                      MMIO

**Trace Types**                      No types assigned.

**Traced Parameters**

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 647 (0X0287)

<u>Description</u>	MIO_MemOpen_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	647 (0X0287)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

# Multi-Media Extensions Major Code: 0X006D Minor Code: 648 (0X0288)

<u>Description</u>	MIO_MemRead_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	648 (0X0288)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

# Multi-Media Extensions Major Code: 0X006D Minor Code: 649 (0X0289)



<b><u>Description</u></b>	MIO_MemWrite_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	649 (0X0289)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 650 (0X028A)

<b><u>Description</u></b>	MIO_MemSeek_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	650 (0X028A)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 651 (0X028B)

<b><u>Description</u></b>	MIO_MemClose_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	651 (0X028B)
<b><u>Trace Groups</u></b>	

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 652 (0X028C)

**Description**

MIO\_mmioOpen\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

652 (0X028C)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 653 (0X028D)

**Description**

MIO\_mmioClose\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

653 (0X028D)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 654 (0X028E)

<u>Description</u>	MIO_mmioRead_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	654 (0X028E)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	rc=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 655 (0X028F)

<u>Description</u>	MIO_mmioWrite_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	655 (0X028F)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	rc=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 656 (0X0290)

<b><u>Description</u></b>	MIO_mmioSeek_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	656 (0X0290)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	IFilePos=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 657 (0X0291)

<b><u>Description</u></b>	MIO_mmioFlush_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	657 (0X0291)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 658 (0X0292)

<b><u>Description</u></b>	MIO_mmioAscend_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	658 (0X0292)
<b><u>Trace Groups</u></b>	

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 659 (0X0293)

**Description**

MIO\_mmioDescend\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

659 (0X0293)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 660 (0X0294)

**Description**

MIO\_mmioAdvance\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

660 (0X0294)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 661 (0X0295)

<u>Description</u>	MIO_mmioInstIOProc_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	661 (0X0295)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 662 (0X0296)

<u>Description</u>	MIO_mmioSendMsg_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	662 (0X0296)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 663 (0X0297)

<b><u>Description</u></b>	MIO_mmioAquireSem_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	663 (0X0297)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 664 (0X0298)

<b><u>Description</u></b>	MIO_mmioDiscardSem_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	664 (0X0298)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 665 (0X0299)

<b><u>Description</u></b>	MIO_mmioCreateChunk_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	665 (0X0299)
<b><u>Trace Groups</u></b>	

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 666 (0X029A)

**Description**

MIO\_mmioCFOpen\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

666 (0X029A)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 667 (0X029B)

**Description**

MIO\_mmioCFClose\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

667 (0X029B)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**



---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 668 (0X029C)

<u>Description</u>	MIO_mmioCFRmvShrEnt_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	668 (0X029C)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 669 (0X029D)

<u>Description</u>	MIO_mmioCFAddShrEnt_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	669 (0X029D)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 670 (0X029E)

<b><u>Description</u></b>	MIO_mmioCFSetInfo_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	670 (0X029E)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 671 (0X029F)

<b><u>Description</u></b>	MIO_mmioCFGetInfo_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	671 (0X029F)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 672 (0X02A0)

<b><u>Description</u></b>	MIO_mmioCFOpnTmpElem_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	672 (0X02A0)
<b><u>Trace Groups</u></b>	

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 673 (0X02A1)

**Description**

MIO\_mmioCFCIsTmpElem\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

673 (0X02A1)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 674 (0X02A2)

**Description**

MIO\_mmioCFAddEnt\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

674 (0X02A2)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 675 (0X02A3)

<u>Description</u>	MIO_mmioCFChgEnt_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	675 (0X02A3)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 676 (0X02A4)

<u>Description</u>	MIO_mmioCFCopy_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	676 (0X02A4)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 677 (0X02A5)

<b><u>Description</u></b>	MIO_mmioCFDelEnt_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	677 (0X02A5)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 678 (0X02A6)

<b><u>Description</u></b>	MIO_mmioCFAddElem_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	678 (0X02A6)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 679 (0X02A7)

<b><u>Description</u></b>	MIO_mmioCFFndEnt_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	679 (0X02A7)
<b><u>Trace Groups</u></b>	

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 680 (0X02A8)

**Description**

MIO\_mmiolidentFile\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

680 (0X02A8)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 681 (0X02A9)

**Description**

MIO\_MidiOpen\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

681 (0X02A9)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 682 (0X02AA)

<u>Description</u>	MIO_MidiRead_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	682 (0X02AA)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 683 (0X02AB)

<u>Description</u>	MIO_MidiWrite_Exit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	683 (0X02AB)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 684 (0X02AC)

<b><u>Description</u></b>	MIO_MidiSeek_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	684 (0X02AC)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 685 (0X02AD)

<b><u>Description</u></b>	MIO_MidiClose_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	685 (0X02AD)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

## Multi-Media Extensions Major Code: 0X006D Minor Code: 704 (0X02C0)

<b><u>Description</u></b>	MSH_RdPlayList_Exit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	704 (0X02C0)
<b><u>Trace Groups</u></b>	



MSH

**Trace Types**

No types assigned.

**Traced Parameters**

Operation=%F Operand1=%F Operand2=%F Operand3=%F rc=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 705 (0X02C1)

**Description**

MSH\_WrPlayList\_Exit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

705 (0X02C1)

**Trace Groups**

MSH

**Trace Types**

No types assigned.

**Traced Parameters**

Operation=%F Operand1=%F Operand2=%F Operand3=%F rc=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 720 (0X02D0)

**Description**

SWVR\_GetImageExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

720 (0X02D0)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

---

# Multi-Media Extensions Major Code: 0X006D Minor Code: 721 (0X02D1)

<u>Description</u>	SWVR_ComplIdxFrameExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	721 (0X02D1)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 722 (0X02D2)

<u>Description</u>	SWVR_CompRefrFrameExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	722 (0X02D2)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

# Multi-Media Extensions Major Code: 0X006D Minor Code: 723 (0X02D3)

<u>Description</u>	SWVR_PostBufferFullExit
<u>Tracepoint</u>	

Static trace point in Multi-Media Extensions.

**Minor Code**

723 (0X02D3)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 724 (0X02D4)

**Description**

SWVR\_PostBufferEmptyExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

724 (0X02D4)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 725 (0X02D5)

**Description**

SWVR\_WaitMsgBufferFullExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

725 (0X02D5)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 726 (0X02D6)

<u>Description</u>	SWVR_WaitMsgBufferEmptyExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	726 (0X02D6)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 727 (0X02D7)

<u>Description</u>	SWVR_PostCountErrorExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	727 (0X02D7)
<u>Trace Groups</u>	MMIO
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	PostCount=%d=%F

-----

Multi-Media Extensions Major Code: 0X006D Minor Code: 728 (0X02D8)

Description

SWVR\_BufferWriteDoneExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

728 (0X02D8)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

ByteCount=%d=%F

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 729 (0X02D9)

**Description**

SWVR\_VCADosDevIOCTLExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

729 (0X02D9)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

---

## Multi-Media Extensions Major Code: 0X006D Minor Code: 730 (0X02DA)

**Description**

SWVR\_VCABufferCopyExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

730 (0X02DA)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 731 (0X02DB)

**Description**

SWVR\_VCAtoX4CopyExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

731 (0X02DB)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 732 (0X02DC)

**Description**

SWVR\_VCAtoX2CopyExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

732 (0X02DC)

**Trace Groups**

MMIO

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 733 (0X02DD)

<b><u>Description</u></b>	SWVR_VCA640CopyExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	733 (0X02DD)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 740 (0X02E4)

<b><u>Description</u></b>	SVSH_DecompBufferExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	740 (0X02E4)
<b><u>Trace Groups</u></b>	MMIO
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	hStream=%F rc=%F BufLen=%F BlockIntvl=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 741 (0X02E5)

<b><u>Description</u></b>	ULDC_DLLLDroppedExit
<b><u>Tracepoint</u></b>	Static trace point in Multi-Media Extensions.
<b><u>Minor Code</u></b>	741 (0X02E5)

**Trace Groups**

SVSHSYNC

**Trace Types**

No types assigned.

**Traced Parameters**

PulseCount=%F ulFrameDiff=%F mmtError=%F fWait=%F fDropFrame=%F

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 745 (0X02E9)

**Description**

SVMC\_SetBitmapExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

745 (0X02E9)

**Trace Groups**

SVSHSYNC

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----

## Multi-Media Extensions Major Code: 0X006D Minor Code: 746 (0X02EA)

**Description**

SVMC\_BitBlitExit

**Tracepoint**

Static trace point in Multi-Media Extensions.

**Minor Code**

746 (0X02EA)

**Trace Groups**

SVSHSYNC

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

-----



# Multi-Media Extensions Major Code: 0X006D Minor Code: 747 (0X02EB)

<u>Description</u>	SVMC_BlitUpExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	747 (0X02EB)
<u>Trace Groups</u>	SVSHSYNC
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

# Multi-Media Extensions Major Code: 0X006D Minor Code: 748 (0X02EC)

<u>Description</u>	SVMC_SpiMoveCoorExit
<u>Tracepoint</u>	Static trace point in Multi-Media Extensions.
<u>Minor Code</u>	748 (0X02EC)
<u>Trace Groups</u>	SVSHSYNC
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	No parameters traced.

## PMSHAPI.DLL Trace Events

The tracepoints for the PMSHAPI.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for PMSHAPI Major Code: 0X00C0, sorted by minor code.

Trace events for PMSHAPI Major Code: 0X00C0 ,sorted by tracepoint.

## Trace Events for PMSHAPI Major Code: 0X00C0, Sorted by Minor Code

00001 (0X0001) ShePIInitialise Pre-Invocation  
00002 (0X0002) WinQueryProfileInt Pre-Invocation  
00003 (0X0003) WinQueryProfileString Pre-Invocation  
00004 (0X0004) WinWriteProfileString Pre-Invocation  
00005 (0X0005) WinQueryProfileSize Pre-Invocation  
00006 (0X0006) WinQueryProfileData Pre-Invocation  
00007 (0X0007) WinWriteProfileData Pre-Invocation  
00008 (0X0008) WinInitSessionMgr Pre-Invocation  
00009 (0X0009) Win16SetFgndWindow Pre-Invocation  
00018 (0X0012) WinAddProgram Pre-Invocation  
00019 (0X0013) WinRemoveProgram Pre-Invocation  
00020 (0X0014) WinChangeProgram Pre-Invocation  
00021 (0X0015) WinQueryDefinition Pre-Invocation  
00022 (0X0016) WinQueryProgramTitles Pre-Invocation  
00023 (0X0017) WinCreateGroup Pre-Invocation  
00024 (0X0018) WinInitializePL Pre-Invocation  
00032 (0X0020) WinQueryProgramUse Pre-Invocation  
00035 (0X0023) WinDestroyGroup Pre-Invocation  
00037 (0X0025) ShlLoadFont Pre-Invocation  
00038 (0X0026) WinQueryProgramType Pre-Invocation  
00039 (0X0027) EntryProc Pre-Invocation  
00040 (0X0028) PMExecRegister Pre-Invocation  
00041 (0X0029) ExitProc Pre-Invocation  
00048 (0X0030) InitMinimizeIcon Pre-Invocation  
00051 (0X0033) WSHInit Pre-Invocation  
00052 (0X0034) SheVioModeWait Pre-Invocation  
00053 (0X0035) SheVioSavRedrawWait Pre-Invocation  
00054 (0X0036) StartSaveWaitThreads Pre-Invocation  
00055 (0X0037) CreateThreadStack Pre-Invocation  
00056 (0X0038) RemoveOS2INI Pre-Invocation  
00065 (0X0041) ReportNoHardErrors Pre-Invocation  
00066 (0X0042) SwitchToNextSession Pre-Invocation  
00067 (0X0043) ActivateSession Pre-Invocation  
00068 (0X0044) WinInstStartApp Pre-Invocation  
00069 (0X0045) WinTerminateApp Pre-Invocation  
00070 (0X0046) WinCreateSwitchEntry Pre-Invocation  
00071 (0X0047) WinQuerySessionTitle Pre-Invocation  
00072 (0X0048) WinAddSwitchEntry Pre-Invocation  
00073 (0X0049) WinChangeSwitchEntry Pre-Invocation  
00086 (0X0056) WinQuerySwitchEntry Pre-Invocation  
00087 (0X0057) ShelInitializeIniFile Pre-Invocation  
00088 (0X0058) InitialiseIniFile Pre-Invocation  
00089 (0X0059) SetDosWarning Pre-Invocation  
00096 (0X0060) CleanUp Pre-Invocation  
00097 (0X0061) ShellPostMessage Pre-Invocation  
00098 (0X0062) ShellSendMessage Pre-Invocation  
00099 (0X0063) strtrn Pre-Invocation  
00100 (0X0064) BadSwitch Pre-Invocation  
00101 (0X0065) WinQueryTaskTitle Pre-Invocation  
00102 (0X0066) WinQueryTaskSizePos Pre-Invocation  
00103 (0X0067) WinQuerySwitchList Pre-Invocation  
00104 (0X0068) WinRemoveSwitchEntry Pre-Invocation  
00105 (0X0069) WinSwitchToProgram Pre-Invocation  
00112 (0X0070) WinSwitchProgramRegister Pre-Invocation  
00113 (0X0071) FindSwitchEntry Pre-Invocation  
00115 (0X0073) WinEndProgram Pre-Invocation  
00116 (0X0074) WinStopProgram Pre-Invocation  
00117 (0X0075) WinEndWindowSession Pre-Invocation  
00118 (0X0076) lpfnShellWndProc Pre-Invocation

00119 (0X0077) lpfnIconWndProc Pre-Invocation  
00120 (0X0078) WinSwitchToTaskManager Pre-Invocation  
00121 (0X0079) fnBadAppDlgProc Pre-Invocation  
00128 (0X0080) WinSwitchToProgram2 Pre-Invocation  
00129 (0X0081) WinProcessHotKey Pre-Invocation  
00130 (0X0082) WinInitSession Pre-Invocation  
00131 (0X0083) WinEndSession Pre-Invocation  
00132 (0X0084) WinInitSwEntry Pre-Invocation  
00133 (0X0085) WinSetSwEntry Pre-Invocation  
00134 (0X0086) WinQueryExtIldFocus Pre-Invocation  
00135 (0X0087) WinSetExtIldFocus Pre-Invocation  
00136 (0X0088) SheSystemShutdown Pre-Invocation  
00137 (0X0089) Start16SystemExecutables Pre-Invocation  
00144 (0X0090) ShlLoadPublicFonts Pre-Invocation  
00145 (0X0091) WinNoShutdown Pre-Invocation  
00147 (0X0093) WinSetTitle Pre-Invocation  
00148 (0X0094) WinCPLRegister Pre-Invocation  
00149 (0X0095) WinPMFILERegister Pre-Invocation  
00150 (0X0096) InitialiseSessionManager Pre-Invocation  
00151 (0X0097) SetKBDHotKey Pre-Invocation  
00257 (0X0101) PrfQueryProfileSize Pre-Invocation  
00258 (0X0102) PrfOpenProfile Pre-Invocation  
00259 (0X0103) PrfCloseProfile Pre-Invocation  
00260 (0X0104) PrfRemoveProgram Pre-Invocation  
00261 (0X0105) PrfCreateGroup Pre-Invocation  
00262 (0X0106) PrfDestroyGroup Pre-Invocation  
00263 (0X0107) PrfQueryProfile Pre-Invocation  
00264 (0X0108) PrfReset Pre-Invocation  
00265 (0X0109) PrfAddProgram Pre-Invocation  
00272 (0X0110) PrfChangeProgram Pre-Invocation  
00273 (0X0111) PrfQueryDefinition Pre-Invocation  
00274 (0X0112) PrfQueryProgramHandle Pre-Invocation  
00275 (0X0113) PrfQueryProgramTitles Pre-Invocation  
00277 (0X0115) PrfQueryProfileString Pre-Invocation  
00278 (0X0116) PrfWriteProfileString Pre-Invocation  
00280 (0X0118) PrfWriteProfileData Pre-Invocation  
00306 (0X0132) PrfQueryProgramCategory Pre-Invocation  
04375 (0X1117) PrfQueryProfileData Pre-Invocation  
32769 (0X8001) ShePIIInitialise Post-Invocation  
32770 (0X8002) WinQueryProfileInt Post-Invocation  
32771 (0X8003) WinSetFgndWindow Post-invocation  
32772 (0X8004) WinWriteProfileString Post-Invocation  
32773 (0X8005) WinSetFgndWindow Post-invocation  
32774 (0X8006) WinQueryProfileData Post-Invocation  
32775 (0X8007) WinWriteProfileData Post-Invocation  
32776 (0X8008) WinInitSessionMgr Post-invocation  
32777 (0X8009) WinSetFgndWindow Post-invocation  
32786 (0X8012) WinAddProgram Post-Invocation  
32787 (0X8013) WinRemoveProgram Post-Invocation  
32788 (0X8014) WinChangeProgram Post-Invocation  
32789 (0X8015) WinQueryDefinition Post-Invocation  
32790 (0X8016) WinQueryProgramTitles Post-Invocation  
32791 (0X8017) WinCreateGroup Post-Invocation  
32792 (0X8018) WinInitializePL Post-Invocation  
32800 (0X8020) WinQueryProgramUse Post-Invocation  
32803 (0X8023) WinDestroyGroup Post-Invocation  
32805 (0X8025) ShlLoadFont Post-invocation  
32806 (0X8026) WinQueryProgramType Post-Invocation  
32807 (0X8027) EntryProc Post-invocation  
32809 (0X8029) ExitProc Post-invocation  
32816 (0X8030) InitMinimizeIcon Post-invocation  
32819 (0X8033) WSHInit Post-invocation  
32820 (0X8034) SheVioModeWait Post-invocation  
32821 (0X8035) SheVioSavRedrawWait Post-invocation  
32822 (0X8036) StartSaveWaitThreads Post-invocation  
32823 (0X8037) CreateThreadStack Post-invocation  
32824 (0X8038) RemoveOS2INI Post-invocation  
32833 (0X8041) ReportNoHardErrors Post-invocation  
32834 (0X8042) SwitchToNextSession Post-Invocation  
32835 (0X8043) ActivateSession Post-Invocation  
32836 (0X8044) WinInstStartApp Post-Invocation  
32837 (0X8045) WinTerminateApp Post-Invocation  
32838 (0X8046) WinCreateSwitchEntry Post-Invocation

32839 (0X8047) WinQuerySessionTitle Post-Invocation  
32840 (0X8048) WinAddSwitchEntry Post-Invocation  
32841 (0X8049) WinChangeSwitchEntry Post-Invocation  
32854 (0X8056) WinQuerySwitchEntry Post-Invocation  
32855 (0X8057) ShellInitializeIniFile Post-Invocation  
32856 (0X8058) InitializeIniFile Post-Invocation  
32857 (0X8059) SetDosWarning Post-Invocation  
32864 (0X8060) CleanUp Post-Invocation  
32865 (0X8061) ShellPostMessage Post-Invocation  
32866 (0X8062) ShellSendMessage Post-Invocation  
32867 (0X8063) strtrn Post-Invocation  
32868 (0X8064) BadSwitch Post-Invocation  
32869 (0X8065) WinQueryTaskTitle Post-Invocation  
32870 (0X8066) WinQueryTaskSizePos Post-Invocation  
32871 (0X8067) WinQuerySwitchList Post-Invocation  
32872 (0X8068) WinRemoveSwitchEntry Post-Invocation  
32873 (0X8069) WinSwitchToProgram Post-Invocation  
32880 (0X8070) WinSwitchProgramRegister Post-invocation  
32881 (0X8071) FindSwitchEntry Post-Invocation  
32883 (0X8073) WinEndProgram Post-Invocation  
32884 (0X8074) WinStopProgram Post-Invocation  
32885 (0X8075) WinEndWindowSession Post-Invocation  
32886 (0X8076) lpfnShellWndProc Post-invocation  
32887 (0X8077) lpfnIconWndProc Post-invocation  
32888 (0X8078) WinSwitchToTaskManager Post-Invocation  
32889 (0X8079) fnBadAppDlgProc Post-Invocation  
32896 (0X8080) WinSwitchToProgram2 Post-Invocation  
32898 (0X8082) WinInitSession Post-Invocation  
32899 (0X8083) WinEndSession Post-Invocation  
32900 (0X8084) WinInitSwEntry Post-Invocation  
32901 (0X8085) WinSetSwEntry Post-Invocation  
32902 (0X8086) WinQueryExtlDFocus Post-invocation  
32903 (0X8087) WinSetExtlDFocus Post-invocation  
32904 (0X8088) SheSystemShutdown Post-Invocation  
32905 (0X8089) StartSystemExecutables Post-invocation  
32912 (0X8090) Shl16LoadPublicFonts Post-invocation  
32913 (0X8091) WinNoShutdown Post-Invocation  
32915 (0X8093) WinSetTitle Post-Invocation  
32916 (0X8094) WinCPLRegister Post-invocation  
32917 (0X8095) WinPMFILERegister Post-invocation  
32918 (0X8096) InitializeSessionManager Post-Invocation  
32919 (0X8097) SetKBDHotKey Post-Invocation

---

## Trace Events for PMSHAPI Major Code: 0X00C0, Sorted by Tracepoint

ACTIVATESESSION 00067 (0X0043)  
ActivateSession 32835 (0X8043)  
BADSWITCH 00100 (0X0064)  
BadSwitch 32868 (0X8064)  
CLEANUP 00096 (0X0060)  
CREATETHREADSTACK 00055 (0X0037)  
CleanUp 32864 (0X8060)  
CreateThreadStack 32823 (0X8037)  
ENTRYPROC 00039 (0X0027)  
EXITPROC 00041 (0X0029)  
EntryProc 32807 (0X8027)  
ExitProc 32809 (0X8029)  
FINDSWITCHENTRY 00113 (0X0071)  
FNBADAPPDLGPROC 00121 (0X0079)  
FindSwitchEntry 32881 (0X8071)  
INITIALISEINIFILE 00088 (0X0058)  
INITIALISESESSIONMANAGER 00150 (0X0096)  
INITMINIMIZEICON 00048 (0X0030)

InitMinimizeIcon 32816 (0X8030)  
InitialiseIniFile 32856 (0X8058)  
InitialiseSessionManager 32918 (0X8096)  
LPFNICONWNDPROC 00119 (0X0077)  
LPFNSHELLWNDPROC 00118 (0X0076)  
PM16EXECREGISTER 00040 (0X0028)  
PRF32ADDPGRAM 00265 (0X0109)  
PRF32CHANGEPROGRAM 00272 (0X0110)  
PRF32CLOSEPROFILE 00259 (0X0103)  
PRF32CREATEGROUP 00261 (0X0105)  
PRF32DESTROYGROUP 00262 (0X0106)  
PRF32OPENPROFILE 00258 (0X0102)  
PRF32QUERYDEFINITION 00273 (0X0111)  
PRF32QUERYPROFILE 00263 (0X0107)  
PRF32QUERYPROFILEDATA 04375 (0X1117)  
PRF32QUERYPROFILESIZE 00257 (0X0101)  
PRF32QUERYPROFILESTRING 00277 (0X0115)  
PRF32QUERYPROGRAMCATEGORY 00306 (0X0132)  
PRF32QUERYPROGRAMHANDLE 00274 (0X0112)  
PRF32QUERYPROGRAMTITLES 00275 (0X0113)  
PRF32REMOVEPROGRAM 00260 (0X0104)  
PRF32RESET 00264 (0X0108)  
PRF32WRITEPROFILEDATA 00280 (0X0118)  
PRF32WRITEPROFILESTRING 00278 (0X0116)  
REMOVEOS2INI 00056 (0X0038)  
REPORTNOHARDERRORS 00065 (0X0041)  
RemoveOS2INI 32824 (0X8038)  
ReportNoHardErrors 32833 (0X8041)  
SETDOSWARNING 00089 (0X0059)  
SETKBDHOTKEY 00151 (0X0097)  
SHE16INITIALIZEINIFILE 00087 (0X0057)  
SHE16PIINITIALISE 00001 (0X0001)  
SHE16SYSTEMSHUTDOWN 00136 (0X0088)  
SHELLPOSTMESSAGE 00097 (0X0061)  
SHELLSENDMESSAGE 00098 (0X0062)  
SHEVIOMODEWAIT 00052 (0X0034)  
SHEVIOSAVREDRAWWAIT 00053 (0X0035)  
SHL16LOADPUBLICFONTS 00144 (0X0090)  
SHLLOADFONT 00037 (0X0025)  
START16SYSTEMEXECUTABLES 00137 (0X0089)  
STARTSAVEWAITTHREADS 00054 (0X0036)  
STRTRN 00099 (0X0063)  
SWITCHTONEXTSESSION 00066 (0X0042)  
SetDosWarning 32857 (0X8059)  
SetKBDHotKey 32919 (0X8097)  
She16InitializeIniFile 32855 (0X8057)  
She16PIInitialise 32769 (0X8001)  
She16SystemShutdown 32904 (0X8088)  
SheVioModeWait 32820 (0X8034)  
SheVioSavRedrawWait 32821 (0X8035)  
ShellPostMessage 32865 (0X8061)  
ShellSendMessage 32866 (0X8062)  
Shl16LoadPublicFonts 32912 (0X8090)  
ShlLoadFont 32805 (0X8025)  
Start16SystemExecutables 32905 (0X8089)  
StartSaveWaitThreads 32822 (0X8036)  
SwitchToNextSession 32834 (0X8042)  
WIN16ADDPGRAM 00018 (0X0012)  
WIN16ADDSWITCHENTRY 00072 (0X0048)  
WIN16CHANGEPROGRAM 00020 (0X0014)  
WIN16CHANGESWITCHENTRY 00073 (0X0049)  
WIN16CPLREGISTER 00148 (0X0094)  
WIN16CREATEGROUP 00023 (0X0017)  
WIN16CREATESWITCHENTRY 00070 (0X0046)  
WIN16DESTROYGROUP 00035 (0X0023)  
WIN16ENDPROGRAM 00115 (0X0073)  
WIN16INITIALIZEPL 00024 (0X0018)  
WIN16INITSESSIONMGR 00008 (0X0008)  
WIN16INSTSTARTAPP 00068 (0X0044)  
WIN16NOSHUTDOWN 00145 (0X0091)  
WIN16PMFILEREREGISTER 00149 (0X0095)  
WIN16QUERYDEFINITION 00021 (0X0015)  
WIN16QUERYPROFILEDATA 00006 (0X0006)

WIN16QUERYPROFILEINT 00002 (0X0002)  
WIN16QUERYPROFILESIZE 00005 (0X0005)  
WIN16QUERYPROFILESTRING 00003 (0X0003)  
WIN16QUERYPROGRAMTITLES 00022 (0X0016)  
WIN16QUERYPROGRAMTYPE 00038 (0X0026)  
WIN16QUERYPROGRAMUSE 00032 (0X0020)  
WIN16QUERYSESSIONTITLE 00071 (0X0047)  
WIN16QUERYSWITCHENTRY 00086 (0X0056)  
WIN16QUERYSWITCHLIST 00103 (0X0067)  
WIN16QUERYTASKSIZEPOS 00102 (0X0066)  
WIN16QUERYTASKTITLE 00101 (0X0065)  
WIN16REMOVEPROGRAM 00019 (0X0013)  
WIN16REMOVESWITCHENTRY 00104 (0X0068)  
WIN16SETFGNDWINDOW 00009 (0X0009)  
WIN16STOPPROGRAM 00116 (0X0074)  
WIN16SWITCHPROGRAMREGISTER 00112 (0X0070)  
WIN16SWITCHTOPPROGRAM 00105 (0X0069)  
WIN16TERMINATEAPP 00069 (0X0045)  
WIN16WRITEPROFILEDATA 00007 (0X0007)  
WIN16WRITEPROFILESTRING 00004 (0X0004)  
WINENDSESSION 00131 (0X0083)  
WINENDWINDOWSESSION 00117 (0X0075)  
WININITSESSION 00130 (0X0082)  
WININITSWENTRY 00132 (0X0084)  
WINPROCESSHOTKEY 00129 (0X0081)  
WINQUERYEXTIDFOCUS 00134 (0X0086)  
WINSETEXTIDFOCUS 00135 (0X0087)  
WINSETSWENTRY 00133 (0X0085)  
WINSETTITLE 00147 (0X0093)  
WINSWITCHTOPPROGRAM2 00128 (0X0080)  
WINSWITCHTOTASKMANAGER 00120 (0X0078)  
WSHINIT 00051 (0X0033)  
WSHInit 32819 (0X8033)  
Win16AddProgram 32786 (0X8012)  
Win16AddSwitchEntry 32840 (0X8048)  
Win16CPLRegister 32916 (0X8094)  
Win16ChangeProgram 32788 (0X8014)  
Win16ChangeSwitchEntry 32841 (0X8049)  
Win16CreateGroup 32791 (0X8017)  
Win16CreateSwitchEntry 32838 (0X8046)  
Win16DestroyGroup 32803 (0X8023)  
Win16EndProgram 32883 (0X8073)  
Win16InitSessionMgr 32776 (0X8008)  
Win16InitializePL 32792 (0X8018)  
Win16InstStartApp 32836 (0X8044)  
Win16NoShutdown 32913 (0X8091)  
Win16PMFILERegister 32917 (0X8095)  
Win16QueryDefinition 32789 (0X8015)  
Win16QueryProfileData 32774 (0X8006)  
Win16QueryProfileInt 32770 (0X8002)  
Win16QueryProfileSize 32773 (0X8005)  
Win16QueryProfileString 32771 (0X8003)  
Win16QueryProgramTitles 32790 (0X8016)  
Win16QueryProgramType 32806 (0X8026)  
Win16QueryProgramUse 32800 (0X8020)  
Win16QuerySessionTitle 32839 (0X8047)  
Win16QuerySwitchEntry 32854 (0X8056)  
Win16QuerySwitchList 32871 (0X8067)  
Win16QueryTaskSizePos 32870 (0X8066)  
Win16QueryTaskTitle 32869 (0X8065)  
Win16RemoveProgram 32787 (0X8013)  
Win16RemoveSwitchEntry 32872 (0X8068)  
Win16SetFgndWindow 32777 (0X8009)  
Win16StopProgram 32884 (0X8074)  
Win16SwitchProgramRegister 32880 (0X8070)  
Win16SwitchToProgram 32873 (0X8069)  
Win16TerminateApp 32837 (0X8045)  
Win16WriteProfileData 32775 (0X8007)  
Win16WriteProfileString 32772 (0X8004)  
WinEndSession 32899 (0X8083)  
WinEndWindowSession 32885 (0X8075)  
WinInitSession 32898 (0X8082)  
WinInitSwEntry 32900 (0X8084)

WinQueryExtIdFocus 32902 (0X8086)  
WinSetExtIdFocus 32903 (0X8087)  
WinSetSwEntry 32901 (0X8085)  
WinSetTitle 32915 (0X8093)  
WinSwitchToProgram2 32896 (0X8080)  
WinSwitchToTaskManager 32888 (0X8078)  
fnBadAppDlgProc 32889 (0X8079)  
lpfnIconWndProc 32887 (0X8077)  
lpfnShellWndProc 32886 (0X8076)  
strtrn 32867 (0X8063)

---

## PMSHAPI Major Code: 0X00C0 Minor Code: 1 (0X0001)

### Description

ShePIIInitialise Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.SHE16PIINITIALISE

### Minor Code

1 (0X0001)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

pPILocation = %a

pPILocation -> %s

---

## PMSHAPI Major Code: 0X00C0 Minor Code: 2 (0X0002)

### Description

WinQueryProfileInt Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROFILEINT

### Minor Code

2 (0X0002)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

Default = %w, pKeyName = %a

pAppName = %a, hab = %a

pKeyName -> %s

pAppName -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 3 (0X0003)

### Description

WinQueryProfileString Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROFILESTRING

### Minor Code

3 (0X0003)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

BuffSize = %w, pBuffer = %a

pDefault = %a, pKeyName = %a

pAppName = %a, hab = %a

pDefault -> %s

pKeyName -> %s

pAppName -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 4 (0X0004)

### Description

WinWriteProfileString Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16WRITEPROFILESTRING

### Minor Code

4 (0X0004)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

pASCIIZ = %a, pKeyName = %a



pAppName = %a, hab = %a  
pASCIIZ -> %s  
pKeyName -> %s  
pAppName -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 5 (0X0005)

### Description

WinQueryProfileSize Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROFILESIZE

### Minor Code

5 (0X0005)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

pSize = %a, pKeyName = %a  
pAppName = %a, hab = %a  
pKeyName -> %s  
pAppName -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 6 (0X0006)

### Description

WinQueryProfileData Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROFILEDATA

### Minor Code

6 (0X0006)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

pBufferSize = %a, pBuffer = %a, pKeyName = %a

pAppName = %a, hab = %a  
pBufferSize -> %w  
pKeyName -> %s  
pAppName -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 7 (0X0007)

### Description

WinWriteProfileData Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16WRITEPROFILEDATA

### Minor Code

7 (0X0007)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

Length = %w, pData = %a, pKeyName = %a  
pAppName = %a, hab = %a  
pKeyName -> %s  
pAppName -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 8 (0X0008)

### Description

WinInitSessionMgr Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16INITSESSIONMGR

### Minor Code

8 (0X0008)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

Return Address = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 9 (0X0009)

<u>Description</u>	Win16SetFgndWindow Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16SETFGNDWINDOW
<u>Minor Code</u>	9 (0X0009)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	sgid = %w, hwndfocus=%d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 18 (0X0012)

<u>Description</u>	WinAddProgram Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16ADDPROGRAM
<u>Minor Code</u>	18 (0X0012)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	GrpHandle = %a, BufPtr = %a, hab = %a progt -> %w, Title/Icon/Exec = %u

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 19 (0X0013)

<u>Description</u>	WinRemoveProgram Pre-Invocation
--------------------	---------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16REMOVEPROGRAM
<u>Minor Code</u>	19 (0X0013)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	PrgHandle = %a

## PMSHAPI Major Code: 0X00C0 Minor Code: 20 (0X0014)

<u>Description</u>	WinChangeProgram Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CHANGEPROGRAM
<u>Minor Code</u>	20 (0X0014)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	BufPtr = %a, TrgHandle = %a
	progt -> %w, Title/Icon/Exec = %u

## PMSHAPI Major Code: 0X00C0 Minor Code: 21 (0X0015)

<u>Description</u>	WinQueryDefinition Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYDEFINITION
<u>Minor Code</u>	21 (0X0015)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE

**Traced Parameters**

InLength = %w, pibble = %a  
TargetHandle = %a, hab = %a

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 22 (0X0016)

**Description**

ExecuteStartUp Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.EXECUTESTARTUP

**Minor Code**

22 (0X0016)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

%Return Address = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 22 (0X0016)

**Description**

WinQueryProgramTitles Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROGRAMTITLES

**Minor Code**

22 (0X0016)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

pTotal = %a, cbBuffer = %w, pBuffer = %a  
TargetHandle = %a, hab = %a

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 23 (0X0017)

<u>Description</u>	WinCreateGroup Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CREATEGROUP
<u>Minor Code</u>	23 (0X0017)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Rhlp = %d, Rth = %d, vis = %w, InString = %a hab = %a InString -> %s

## PMSHAPI Major Code: 0X00C0 Minor Code: 24 (0X0018)

<u>Description</u>	WinInitializePL Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16INITIALIZEPL
<u>Minor Code</u>	24 (0X0018)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	flOptions=%F

## PMSHAPI Major Code: 0X00C0 Minor Code: 32 (0X0020)

<u>Description</u>	WinQueryProgramUse Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROGRAMUSE
<u>Minor Code</u>	

32 (0X0020)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

Exe = %a

Exe -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 35 (0X0023)

**Description**

WinDestroyGroup Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16DESTROYGROUP

**Minor Code**

35 (0X0023)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

GrpHandle = %a

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 37 (0X0025)

**Description**

ShlLoadFont Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.SHLOADFONT

**Minor Code**

37 (0X0025)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

pFontName -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 38 (0X0026)

<u>Description</u>	WinQueryProgramType Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYPROGRAMTYPE
<u>Minor Code</u>	38 (0X0026)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	Exe = %a
	Exe -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 39 (0X0027)

<u>Description</u>	EntryProc Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.ENTRYPROC
<u>Minor Code</u>	39 (0X0027)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	fForeGround = %w, hswitch = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 40 (0X0028)

<u>Description</u>	PMExecRegister Pre-Invocation
--------------------	-------------------------------



<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.PM16EXECREGISTER
<u>Minor Code</u>	40 (0X0028)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hwnd = %a

PMSHAPI Major Code: 0X00C0 Minor Code: 41 (0X0029)

<u>Description</u>	ExitProc Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.EXITPROC
<u>Minor Code</u>	41 (0X0029)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	mp2 = %d, mp1 = %d, message = %w, hwnd = %d

PMSHAPI Major Code: 0X00C0 Minor Code: 48 (0X0030)

<u>Description</u>	InitMinimizeIcon Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.INITMINIMIZEICON
<u>Minor Code</u>	48 (0X0030)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

pswctl = %w, pszIconFile = %a  
pszIconFile -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 51 (0X0033)

### Description

WSHInit Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WSHINIT

### Minor Code

51 (0X0033)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

pStack\_AR = %a

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 52 (0X0034)

### Description

SheVioModeWait Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.SHEVIOMODEWAIT

### Minor Code

52 (0X0034)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

Return Address = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 53 (0X0035)

<u>Description</u>	SheVioSavRedrawWait Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.SHEVIOSAVREDRAWWAIT
<u>Minor Code</u>	53 (0X0035)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Return Address = %w

## PMSHAPI Major Code: 0X00C0 Minor Code: 54 (0X0036)

<u>Description</u>	StartSaveWaitThreads Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.STARTSAVEWAITTTHREADS
<u>Minor Code</u>	54 (0X0036)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Return Address = %w

## PMSHAPI Major Code: 0X00C0 Minor Code: 55 (0X0037)

<u>Description</u>	CreateThreadStack Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.CREATETHREADSTACK
<u>Minor Code</u>	55 (0X0037)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE

**Traced Parameters**

StackSize = %w

-----

**PMSHAPI Major Code: 0X00C0 Minor Code: 56 (0X0038)**

**Description**

RemoveOS2INI Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.REMOVEOS2INI

**Minor Code**

56 (0X0038)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

Return Address = %w

-----

**PMSHAPI Major Code: 0X00C0 Minor Code: 65 (0X0041)**

**Description**

ReportNoHardErrors Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.REPORTNOHARDERRORS

**Minor Code**

65 (0X0041)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

pszTitle = %a, hwnd = %d

pszTitle -> %s

-----

**PMSHAPI Major Code: 0X00C0 Minor Code: 66 (0X0042)**

<b>Description</b>	SwitchToNextSession Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMSHAPI.SWITCHTONEXTSESSION
<b>Minor Code</b>	66 (0X0042)
<b>Trace Groups</b>	SHAPI
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	fPrev = %w, fsIn = %w

## PMSHAPI Major Code: 0X00C0 Minor Code: 67 (0X0043)

<b>Description</b>	ActivateSession Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMSHAPI.ACTIVATESESSION
<b>Minor Code</b>	67 (0X0043)
<b>Trace Groups</b>	SHAPI
<b>Trace Types</b>	PRE
<b>Traced Parameters</b>	fs = %w, hwndToBottom = %d, pswi = %w

## PMSHAPI Major Code: 0X00C0 Minor Code: 68 (0X0044)

<b>Description</b>	WinInstStartApp Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16INSTSTARTAPP
<b>Minor Code</b>	68 (0X0044)
<b>Trace Groups</b>	SHAPI

**Trace Types** PRE

**Traced Parameters**

fsOptions = %w, pData = %a, pszCmdLine = %a, aszApplication = %a  
cCount = %w, hwndNotifyWindow = %d, hini = %d

PMSHAPI Major Code: 0X00C0 Minor Code: 69 (0X0045)

**Description** WinTerminateApp Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.WIN16TERMINATEAPP

**Minor Code** 69 (0X0045)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

happ = %d %d

PMSHAPI Major Code: 0X00C0 Minor Code: 70 (0X0046)

**Description** WinCreateSwitchEntry Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CREATESWITCHENTRY

**Minor Code** 70 (0X0046)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

lpswctl = %a, hablgnored = %w

# PMSHAPI Major Code: 0X00C0 Minor Code: 71 (0X0047)

Description	WinQuerySessionTitle Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYSESSIONTITLE
Minor Code	71 (0X0047)
Trace Groups	SHAPI
Trace Types	PRE
Traced Parameters	cbNameBufferLength = %d, szNameBuffer = %a, sid = %w, szNameBuffer -> %s

# PMSHAPI Major Code: 0X00C0 Minor Code: 72 (0X0048)

Description	WinAddSwitchEntry Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16ADDSWITCHENTRY
Minor Code	72 (0X0048)
Trace Groups	SHAPI
Trace Types	PRE
Traced Parameters	lpswctl = %a

# PMSHAPI Major Code: 0X00C0 Minor Code: 73 (0X0049)

Description	WinChangeSwitchEntry Pre-Invocation
Tracepoint	

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CHANGESWITCHENTRY

**Minor Code**

73 (0X0049)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

lpswctl = %a, hswitch = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 86 (0X0056)

**Description**

WinQuerySwitchEntry Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYSWITCHENTRY

**Minor Code**

86 (0X0056)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

lpswctl = %a, hswitch = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 87 (0X0057)

**Description**

ShelInitializeIniFile Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.SHE16INITIALIZEINIFILE

**Minor Code**

87 (0X0057)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**



Return Address = %w

---

## PMSHAPI Major Code: 0X00C0 Minor Code: 88 (0X0058)

### Description

InitialiseIniFile Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.INITIALISEINIFILE

### Minor Code

88 (0X0058)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

Return Address = %w

---

## PMSHAPI Major Code: 0X00C0 Minor Code: 89 (0X0059)

### Description

SetDosWarning Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.SETDOSWARNING

### Minor Code

89 (0X0059)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

Rc = %w

---

## PMSHAPI Major Code: 0X00C0 Minor Code: 96 (0X0060)

### Description

CleanUp Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSHAPI.CLEANUP
<b><u>Minor Code</u></b>	96 (0X0060)
<b><u>Trace Groups</u></b>	SHAPI
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	Rc = %w

## PMSHAPI Major Code: 0X00C0 Minor Code: 97 (0X0061)

<b><u>Description</u></b>	ShellPostMessage Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSHAPI.SHELLPOSTMESSAGE
<b><u>Minor Code</u></b>	97 (0X0061)
<b><u>Trace Groups</u></b>	SHAPI
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	pswi = %w, mp1 = %d

## PMSHAPI Major Code: 0X00C0 Minor Code: 98 (0X0062)

<b><u>Description</u></b>	ShellSendMessage Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSHAPI.SHELLSENDMESSAGE
<b><u>Minor Code</u></b>	98 (0X0062)
<b><u>Trace Groups</u></b>	SHAPI
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	

pswi = %d, mp1 = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 99 (0X0063)

### Description

strtrn Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.STRTRN

### Minor Code

99 (0X0063)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

cch = %w, pszTo = %a, pszFrom = %a

pszTo -> %s

pszFrom -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 100 (0X0064)

### Description

BadSwitch Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.BADSWITCH

### Minor Code

100 (0X0064)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

hswitch = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 101 (0X0065)

<u>Description</u>	WinQueryTaskTitle Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYTASKTITLE
<u>Minor Code</u>	101 (0X0065)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	cbNameBufferLength = %w, szNameBuffer = %a, sid = %w szNameBuffer -> %s

## PMSHAPI Major Code: 0X00C0 Minor Code: 102 (0X0066)

<u>Description</u>	WinQueryTaskSizePos Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYTASKSIZEPOS
<u>Minor Code</u>	102 (0X0066)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	pswp = %a, sid = %w, hablgnored = %w

## PMSHAPI Major Code: 0X00C0 Minor Code: 103 (0X0067)

<u>Description</u>	WinQuerySwitchList Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16QUERYSWITCHLIST
<u>Minor Code</u>	103 (0X0067)

**Trace Groups**  
SHAPI

**Trace Types**  
PRE

**Traced Parameters**  
  
cbBufferLength = %w, pswBlock = %a,

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 104 (0X0068)

**Description**  
WinRemoveSwitchEntry Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16REMOVESWITCHENTRY

**Minor Code**  
104 (0X0068)

**Trace Groups**  
SHAPI

**Trace Types**  
PRE

**Traced Parameters**  
  
hswitch = %d

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 105 (0X0069)

**Description**  
WinSwitchToProgram Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.WIN16SWITCHTOPROGRAM

**Minor Code**  
105 (0X0069)

**Trace Groups**  
SHAPI

**Trace Types**  
PRE

**Traced Parameters**  
  
hswitch = %d

-----

# PMSHAPI Major Code: 0X00C0 Minor Code: 112 (0X0070)

Description	WinSwitchProgramRegister Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16SWITCHPROGRAMREGISTER
Minor Code	112 (0X0070)
Trace Groups	SHAPI
Trace Types	PRE
Traced Parameters	<p>pfnwp = %a, hmq = %d, hwnd = %d</p> <p>-----</p>

# PMSHAPI Major Code: 0X00C0 Minor Code: 113 (0X0071)

Description	FindSwitchEntry Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSHAPI.FINDSWITCHENTRY
Minor Code	113 (0X0071)
Trace Groups	SHAPI
Trace Types	PRE
Traced Parameters	<p>fs = %w, sid = %w</p> <p>-----</p>

# PMSHAPI Major Code: 0X00C0 Minor Code: 115 (0X0073)

Description	WinEndProgram Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSHAPI.WIN16ENDPROGRAM
Minor Code	

115 (0X0073)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

idProcess = %w, hswitch = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 116 (0X0074)

**Description**

WinStopProgram Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16STOPPROGRAM

**Minor Code**

116 (0X0074)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

idProcess = %w, hswitch = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 117 (0X0075)

**Description**

WinEndWindowSession Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WINENDWINDOWSESSION

**Minor Code**

117 (0X0075)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hwnd = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 118 (0X0076)

<u>Description</u>	lpfnShellWndProc Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.LPFNSHELLWNDPROC
<u>Minor Code</u>	118 (0X0076)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	mp2 = %d, mp1 = %d, message = %w, hwnd = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 119 (0X0077)

<u>Description</u>	lpfnIconWndProc Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.LPFNICONWNDPROC
<u>Minor Code</u>	119 (0X0077)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	mp2 = %d, mp1 = %d, message = %w, hwnd = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 120 (0X0078)

<u>Description</u>	WinSwitchToTaskManager Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WINSWITCHTOTASKMANAGER



**Minor Code** 120 (0X0078)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**  
Return Address = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 121 (0X0079)

**Description** fnBadAppDlgProc Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.FNBADAPPDLGPROC

**Minor Code** 121 (0X0079)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**  
mp2 = %d, mp1 = %d, message = %w, hwnd = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 128 (0X0080)

**Description** WinSwitchToProgram2 Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.WINSWITCHTOPROGRAM2

**Minor Code** 128 (0X0080)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**  
fs = %w, hswitch = %d

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 129 (0X0081)

<u>Description</u>	WinProcessHotKey Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WINPROCESSHOTKEY
<u>Minor Code</u>	129 (0X0081)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	fProcess = %w, pqmsg = %a

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 130 (0X0082)

<u>Description</u>	WinInitSession Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WININITSESSION
<u>Minor Code</u>	130 (0X0082)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	pReqBlock = %a, ppfn = %a

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 131 (0X0083)

<u>Description</u>	WinEndSession Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WINENDSESSION

**Minor Code** 131 (0X0083)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

sidEnded = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 132 (0X0084)

**Description** WinInitSwEntry Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.WININITSWENTRY

**Minor Code** 132 (0X0084)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

Return Address = %w

PMSHAPI Major Code: 0X00C0 Minor Code: 133 (0X0085)

**Description** WinSetSwEntry Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.WINSETSWENTRY

**Minor Code** 133 (0X0085)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

pReqBlock = %a, ppfn = %a

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 134 (0X0086)

<u>Description</u>	WinQueryExtIldFocus Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WINQUERYEXTIDFOCUS
<u>Minor Code</u>	134 (0X0086)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	pSessionId = %F

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 135 (0X0087)

<u>Description</u>	WinSetExtIldFocus Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WINSETEXTIDFOCUS
<u>Minor Code</u>	135 (0X0087)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	
	hSwitch = %d
	sid = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 136 (0X0088)

<u>Description</u>	SheSystemShutdown Pre-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: PMSHAPI.SHE16SYSTEMSHUTDOWN

**Minor Code**

136 (0X0088)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

Return Address = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 137 (0X0089)

**Description**

Start16SystemExecutables Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.START16SYSTEMEXECUTABLES

**Minor Code**

137 (0X0089)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

%Return Address = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 144 (0X0090)

**Description**

ShlLoadPublicFonts Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.SHL16LOADPUBLICFONTS

**Minor Code**

144 (0X0090)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

pszFontDLL -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 145 (0X0091)

### Description

WinNoShutdown Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16NOSHUTDOWN

### Minor Code

145 (0X0091)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

flag = %w, sid = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 147 (0X0093)

### Description

WinSetTitle Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.WINSETTITLE

### Minor Code

147 (0X0093)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

pNewTitle = %a

pNewTitle -> %s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 148 (0X0094)

### Description

WinCPLRegister Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16CPLREGISTER

**Minor Code**

148 (0X0094)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hwnd = %d

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 149 (0X0095)

**Description**

WinPMFILERegister Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WIN16PMFILEREREGISTER

**Minor Code**

149 (0X0095)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hwnd = %d

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 150 (0X0096)

**Description**

InitialiseSessionManager Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.INITIALISESESSIONMANAGER

**Minor Code**

150 (0X0096)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

Return Address = %w

-----

**PMSHAPI Major Code: 0X00C0 Minor Code: 151 (0X0097)**

**Description**

SetKBDHotKey Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.SETKBDHOTKEY

**Minor Code**

151 (0X0097)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

Return Address = %w

-----

**PMSHAPI Major Code: 0X00C0 Minor Code: 257 (0X0101)**

**Description**

PrfQueryProfileSize Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILESIZE

**Minor Code**

257 (0X0101)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hini=%F, pszApp =%f, pszKey=%f, pulReqLen=%F

App=%s, Key=%s

-----

**PMSHAPI Major Code: 0X00C0 Minor Code: 258 (0X0102)**



<u>Description</u>	PrfOpenProfile Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32OPENPROFILE
<u>Minor Code</u>	258 (0X0102)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hab =%F, pszFileName=%f, FileName=%s -----

## PMSHAPI Major Code: 0X00C0 Minor Code: 259 (0X0103)

<u>Description</u>	PrfCloseProfile Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32CLOSEPROFILE
<u>Minor Code</u>	259 (0X0103)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hini = %F -----

## PMSHAPI Major Code: 0X00C0 Minor Code: 260 (0X0104)

<u>Description</u>	PrfRemoveProgram Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32REMOVEPROGRAM
<u>Minor Code</u>	260 (0X0104)
<u>Trace Groups</u>	SHAPI

**Trace Types**

PRE

**Traced Parameters**

hini=%F, ProgHandle=%F

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 261 (0X0105)

**Description**

PrfCreateGroup Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32CREATEGROUP

**Minor Code**

261 (0X0105)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hini=%F, pszTitle=%F, chVisibility=%b, Title=%s

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 262 (0X0106)

**Description**

PrfDestroyGroup Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32DESTROYGROUP

**Minor Code**

262 (0X0106)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hini = %F, hprogGroup=%F

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 263 (0X0107)

<u><b>Description</b></u>	PrfQueryProfile Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILE
<u><b>Minor Code</b></u>	263 (0X0107)
<u><b>Trace Groups</b></u>	SHAPI
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	hab=%F, pPrfProfile=%f

## PMSHAPI Major Code: 0X00C0 Minor Code: 264 (0X0108)

<u><b>Description</b></u>	PrfReset Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32RESET
<u><b>Minor Code</b></u>	264 (0X0108)
<u><b>Trace Groups</b></u>	SHAPI
<u><b>Trace Types</b></u>	PRE
<u><b>Traced Parameters</b></u>	hab      =%f, pPrfProfile=%f cchUserName=%F, pszUserName=%F cchSysName =%F, pszSysName=%F UserName=%s, SysName=%s

## PMSHAPI Major Code: 0X00C0 Minor Code: 265 (0X0109)

<u><b>Description</b></u>	PrfAddProgram Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32ADDPROGRAM

**Minor Code** 265 (0X0109)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

hini        =%F, pDetails    =%F, hprogGroup    =%F

pDetails.length=%F, progt.category=%F, progt.fbVisible=%F

pszTitle    =%F, pszExecutable=%F, pszParameters =%F

pszStartupDir=%F, pszIcon    =%F, pszEnvironment =%F

swpInnit.fl    =%F, cy        =%F, cx        =%F

y            =%F, x        =%F, hwndInsertB    =%F

hwnd        =%F, ulRes1    =%F, ulRes2        =%F

Title    =%s, Executable=%s, Parameters=%s

StartupDir=%s, Icon=%s, Environment=%s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 272 (0X0110)

**Description** PrfChangeProgram Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.PRF32CHANGEPROGRAM

**Minor Code** 272 (0X0110)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

hini        =%F, pDetails    =%F, hprogGroup    =%F

pDetails.length=%F, progt.category=%F, progt.fbVisible=%F

pszTitle    =%F, pszExecutable=%F, pszParameters =%F

pszStartupDir=%F, pszIcon    =%F, pszEnvironment =%F

swpInnit.fl    =%F, cy        =%F, cx        =%F

y            =%F, x        =%F, hwndInsertB    =%F

hwnd        =%F, ulRes1    =%F, ulRes2        =%F

Title =%s, Executable=%s, Parameters=%s  
StartupDir=%s, Icon=%s, Environment=%s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 273 (0X0111)

### Description

PrfQueryDefinition Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYDEFINITION

### Minor Code

273 (0X0111)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

hini=%F, hprog=%F, pDetails=%F, cchBufferMax=%F

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 274 (0X0112)

### Description

PrfQueryProgramHandle Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROGRAMHANDLE

### Minor Code

274 (0X0112)

### Trace Groups

SHAPI

### Trace Types

PRE

### Traced Parameters

hini=%F, pszExeName=%F, phpga=%F, cb=%F  
pcHandles=%F, ExeName=%s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 275 (0X0113)

<u>Description</u>	PrfQueryProgramTitles Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROGRAMTITLES
<u>Minor Code</u>	275 (0X0113)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	<div>hini = %F, hprogGroup = %F, pTitles = %F</div> <div>cchBufferMax = %F, pulCount = %F</div>

## PMSHAPI Major Code: 0X00C0 Minor Code: 277 (0X0115)

<u>Description</u>	PrfQueryProfileString Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILESTRING
<u>Minor Code</u>	277 (0X0115)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	<div>hini =%F, pszApp =%f, pszKey =%f</div> <div>pszDefault =%f, pBuffer=%f, cchBufferMax=%F</div> <div>App=%s, Key=%s, Default=%s</div>

## PMSHAPI Major Code: 0X00C0 Minor Code: 278 (0X0116)

<u>Description</u>	PrfWriteProfileString Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.PRF32WRITEPROFILESTRING
<u>Minor Code</u>	278 (0X0116)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hini=%F, pszApp=%f, pszKey=%f, pszData=%F

App=%s

Key=%s

Data=%s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 280 (0X0118)

**Description**

PrfWriteProfileData Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32WRITEPROFILEDATA

**Minor Code**

280 (0X0118)

**Trace Groups**

SHAPI

**Trace Types**

PRE

**Traced Parameters**

hini=%F, pszApp=%f, pszKey=%f, pBuf=%F, cchDataLen=%F

App=%s, Key=%s

Data=%u

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 306 (0X0132)

**Description**

PrfQueryProgramCategory Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROGRAMCATEGORY

**Minor Code**

306 (0X0132)

**Trace Groups**

SHAPI

**Trace Types** PRE

**Traced Parameters**

hini=%F, pszExe=%F, Exe=%s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 4375 (0X1117)

**Description** PrfQueryProfileData Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.PRF32QUERYPROFILEDATA

**Minor Code** 4375 (0X1117)

**Trace Groups** SHAPI

**Trace Types** PRE

**Traced Parameters**

hini=%F, pszApp=%f, pszKey=%f, pBuf=%F, pulBuflen=%F

Buflen=%F, App=%s, Key=%s

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32769 (0X8001)

**Description** ShePIInitialise Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSHAPI.She16PIInitialise

**Minor Code** 32769 (0X8001)

**Trace Groups** SHAPI

**Trace Types** POST

**Traced Parameters**

Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32770 (0X8002)



<u><b>Description</b></u>	WinQueryProfileInt Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProfileInt
<u><b>Minor Code</b></u>	32770 (0X8002)
<u><b>Trace Groups</b></u>	SHAPI
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32771 (0X8003)

<u><b>Description</b></u>	WinSetFgndWindow Post-invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProfileString
<u><b>Minor Code</b></u>	32771 (0X8003)
<u><b>Trace Groups</b></u>	SHAPI
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32772 (0X8004)

<u><b>Description</b></u>	WinWriteProfileString Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16WriteProfileString
<u><b>Minor Code</b></u>	32772 (0X8004)
<u><b>Trace Groups</b></u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32773 (0X8005)

Description  
WinSetFgndWindow Post-invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProfileSize

Minor Code  
32773 (0X8005)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32774 (0X8006)

Description  
WinQueryProfileData Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProfileData

Minor Code  
32774 (0X8006)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32775 (0X8007)

<u>Description</u>	WinWriteProfileData Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16WriteProfileData
<u>Minor Code</u>	32775 (0X8007)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32776 (0X8008)

<u>Description</u>	WinInitSessionMgr Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16InitSessionMgr
<u>Minor Code</u>	32776 (0X8008)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32777 (0X8009)

<u>Description</u>	WinSetFgndWindow Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16SetFgndWindow
<u>Minor Code</u>	32777 (0X8009)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32786 (0X8012)

Description  
WinAddProgram Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16AddProgram

Minor Code  
32786 (0X8012)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32787 (0X8013)

Description  
WinRemoveProgram Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16RemoveProgram

Minor Code  
32787 (0X8013)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32788 (0X8014)

<u>Description</u>	WinChangeProgram Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16ChangeProgram
<u>Minor Code</u>	32788 (0X8014)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32789 (0X8015)

<u>Description</u>	WinQueryDefinition Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryDefinition
<u>Minor Code</u>	32789 (0X8015)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32790 (0X8016)

<u>Description</u>	WinQueryProgramTitles Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProgramTitles
<u>Minor Code</u>	32790 (0X8016)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32791 (0X8017)

Description  
WinCreateGroup Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16CreateGroup

Minor Code  
32791 (0X8017)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32792 (0X8018)

Description  
WinInitializePL Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16InitializePL

Minor Code  
32792 (0X8018)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32800 (0X8020)

<u>Description</u>	WinQueryProgramUse Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProgramUse
<u>Minor Code</u>	32800 (0X8020)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32803 (0X8023)

<u>Description</u>	WinDestroyGroup Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16DestroyGroup
<u>Minor Code</u>	32803 (0X8023)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32805 (0X8025)

<u>Description</u>	ShlLoadFont Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.ShlLoadFont
<u>Minor Code</u>	32805 (0X8025)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32806 (0X8026)

Description  
WinQueryProgramType Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryProgramType

Minor Code  
32806 (0X8026)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32807 (0X8027)

Description  
EntryProc Post-invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.EntryProc

Minor Code  
32807 (0X8027)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32809 (0X8029)



<u>Description</u>	ExitProc Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.ExitProc
<u>Minor Code</u>	32809 (0X8029)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32816 (0X8030)

<u>Description</u>	InitMinimizeIcon Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.InitMinimizeIcon
<u>Minor Code</u>	32816 (0X8030)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32819 (0X8033)

<u>Description</u>	WSHInit Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WSHInit
<u>Minor Code</u>	32819 (0X8033)
<u>Trace Groups</u>	SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32820 (0X8034)

**Description**

SheVioModeWait Post-invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.SheVioModeWait

**Minor Code**

32820 (0X8034)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32821 (0X8035)

**Description**

SheVioSavRedrawWait Post-invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.SheVioSavRedrawWait

**Minor Code**

32821 (0X8035)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32822 (0X8036)

<u>Description</u>	StartSaveWaitThreads Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.StartSaveWaitThreads
<u>Minor Code</u>	32822 (0X8036)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32823 (0X8037)

<u>Description</u>	CreateThreadStack Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.CreateThreadStack
<u>Minor Code</u>	32823 (0X8037)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32824 (0X8038)

<u>Description</u>	RemoveOS2INI Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.RemoveOS2INI
<u>Minor Code</u>	32824 (0X8038)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32833 (0X8041)

Description  
ReportNoHardErrors Post-invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.ReportNoHardErrors

Minor Code  
32833 (0X8041)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32834 (0X8042)

Description  
SwitchToNextSession Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.SwitchToNextSession

Minor Code  
32834 (0X8042)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32835 (0X8043)

<u>Description</u>	ActivateSession Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.ActivateSession
<u>Minor Code</u>	32835 (0X8043)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32836 (0X8044)

<u>Description</u>	WinInstStartApp Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16InstStartApp
<u>Minor Code</u>	32836 (0X8044)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32837 (0X8045)

<u>Description</u>	WinTerminateApp Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16TerminateApp
<u>Minor Code</u>	32837 (0X8045)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32838 (0X8046)

Description  
WinCreateSwitchEntry Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16CreateSwitchEntry

Minor Code  
32838 (0X8046)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32839 (0X8047)

Description  
WinQuerySessionTitle Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QuerySessionTitle

Minor Code  
32839 (0X8047)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32840 (0X8048)

<u>Description</u>	WinAddSwitchEntry Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16AddSwitchEntry
<u>Minor Code</u>	32840 (0X8048)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32841 (0X8049)

<u>Description</u>	WinChangeSwitchEntry Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16ChangeSwitchEntry
<u>Minor Code</u>	32841 (0X8049)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32854 (0X8056)

<u>Description</u>	WinQuerySwitchEntry Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QuerySwitchEntry
<u>Minor Code</u>	32854 (0X8056)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32855 (0X8057)

Description  
SheInitializeIniFile Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.She16InitializeIniFile

Minor Code  
32855 (0X8057)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32856 (0X8058)

Description  
InitialiseIniFile Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.InitialiseIniFile

Minor Code  
32856 (0X8058)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32857 (0X8059)



<u>Description</u>	SetDosWarning Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.SetDosWarning
<u>Minor Code</u>	32857 (0X8059)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32864 (0X8060)

<u>Description</u>	CleanUp Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.CleanUp
<u>Minor Code</u>	32864 (0X8060)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32865 (0X8061)

<u>Description</u>	ShellPostMessage Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.ShellPostMessage
<u>Minor Code</u>	32865 (0X8061)
<u>Trace Groups</u>	SHAPI

Trace Types POST

Traced Parameters  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32866 (0X8062)

Description ShellSendMessage Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMSHAPI.ShellSendMessage

Minor Code 32866 (0X8062)

Trace Groups SHAPI

Trace Types POST

Traced Parameters  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32867 (0X8063)

Description strtrn Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMSHAPI.strtrn

Minor Code 32867 (0X8063)

Trace Groups SHAPI

Trace Types POST

Traced Parameters  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32868 (0X8064)

<u>Description</u>	BadSwitch Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.BadSwitch
<u>Minor Code</u>	32868 (0X8064)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32869 (0X8065)

<u>Description</u>	WinQueryTaskTitle Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryTaskTitle
<u>Minor Code</u>	32869 (0X8065)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32870 (0X8066)

<u>Description</u>	WinQueryTaskSizePos Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16QueryTaskSizePos
<u>Minor Code</u>	32870 (0X8066)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32871 (0X8067)

Description  
WinQuerySwitchList Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16QuerySwitchList

Minor Code  
32871 (0X8067)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32872 (0X8068)

Description  
WinRemoveSwitchEntry Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16RemoveSwitchEntry

Minor Code  
32872 (0X8068)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32873 (0X8069)

<u>Description</u>	WinSwitchToProgram Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16SwitchToProgram
<u>Minor Code</u>	32873 (0X8069)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32880 (0X8070)

<u>Description</u>	WinSwitchProgramRegister Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16SwitchProgramRegister
<u>Minor Code</u>	32880 (0X8070)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32881 (0X8071)

<u>Description</u>	FindSwitchEntry Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.FindSwitchEntry
<u>Minor Code</u>	32881 (0X8071)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32883 (0X8073)

Description  
WinEndProgram Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16EndProgram

Minor Code  
32883 (0X8073)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32884 (0X8074)

Description  
WinStopProgram Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Win16StopProgram

Minor Code  
32884 (0X8074)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32885 (0X8075)

<u>Description</u>	WinEndWindowSession Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WinEndWindowSession
<u>Minor Code</u>	32885 (0X8075)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32886 (0X8076)

<u>Description</u>	lpfnShellWndProc Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.lpfnShellWndProc
<u>Minor Code</u>	32886 (0X8076)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32887 (0X8077)

<u>Description</u>	lpfnIconWndProc Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.lpfnIconWndProc
<u>Minor Code</u>	32887 (0X8077)
<u>Trace Groups</u>	SHAPI

**Trace Types**  
POST

**Traced Parameters**  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32888 (0X8078)

**Description**  
WinSwitchToTaskManager Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.WinSwitchToTaskManager

**Minor Code**  
32888 (0X8078)

**Trace Groups**  
SHAPI

**Trace Types**  
POST

**Traced Parameters**  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32889 (0X8079)

**Description**  
fnBadAppDlgProc Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSHAPI.fnBadAppDlgProc

**Minor Code**  
32889 (0X8079)

**Trace Groups**  
SHAPI

**Trace Types**  
POST

**Traced Parameters**  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32896 (0X8080)



<u>Description</u>	WinSwitchToProgram2 Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WinSwitchToProgram2
<u>Minor Code</u>	32896 (0X8080)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32898 (0X8082)

<u>Description</u>	WinInitSession Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WinInitSession
<u>Minor Code</u>	32898 (0X8082)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32899 (0X8083)

<u>Description</u>	WinEndSession Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WinEndSession
<u>Minor Code</u>	32899 (0X8083)
<u>Trace Groups</u>	SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32900 (0X8084)

**Description**

WinInitSwEntry Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WinInitSwEntry

**Minor Code**

32900 (0X8084)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32901 (0X8085)

**Description**

WinSetSwEntry Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.WinSetSwEntry

**Minor Code**

32901 (0X8085)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32902 (0X8086)

<u>Description</u>	WinQueryExtIldFocus Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WinQueryExtIldFocus
<u>Minor Code</u>	32902 (0X8086)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32903 (0X8087)

<u>Description</u>	WinSetExtIldFocus Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WinSetExtIldFocus
<u>Minor Code</u>	32903 (0X8087)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	Return Code = %w

-----

## PMSHAPI Major Code: 0X00C0 Minor Code: 32904 (0X8088)

<u>Description</u>	SheSystemShutdown Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.She16SystemShutdown
<u>Minor Code</u>	32904 (0X8088)
<u>Trace Groups</u>	SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32905 (0X8089)

Description  
StartSystemExecutables Post-invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Start16SystemExecutables

Minor Code  
32905 (0X8089)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32912 (0X8090)

Description  
Shl16LoadPublicFonts Post-invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSHAPI.Shl16LoadPublicFonts

Minor Code  
32912 (0X8090)

Trace Groups  
SHAPI

Trace Types  
POST

Traced Parameters  
  
Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32913 (0X8091)

<u>Description</u>	WinNoShutdown Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16NoShutdown
<u>Minor Code</u>	32913 (0X8091)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32915 (0X8093)

<u>Description</u>	WinSetTitle Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.WinSetTitle
<u>Minor Code</u>	32915 (0X8093)
<u>Trace Groups</u>	SHAPI
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32916 (0X8094)

<u>Description</u>	WinCPLRegister Post-invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSHAPI.Win16CPLRegister
<u>Minor Code</u>	32916 (0X8094)
<u>Trace Groups</u>	SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32917 (0X8095)

**Description**

WinPMFILERegister Post-invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.Win16PMFILERegister

**Minor Code**

32917 (0X8095)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32918 (0X8096)

**Description**

InitialiseSessionManager Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSHAPI.InitialiseSessionManager

**Minor Code**

32918 (0X8096)

**Trace Groups**

SHAPI

**Trace Types**

POST

**Traced Parameters**

Return Code = %w

-----

PMSHAPI Major Code: 0X00C0 Minor Code: 32919 (0X8097)

<b>Description</b>	SetKBDHotKey Post-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMSHAPI.SetKBDHotKey
<b>Minor Code</b>	32919 (0X8097)
<b>Trace Groups</b>	SHAPI
<b>Trace Types</b>	POST
<b>Traced Parameters</b>	
	Return Code = %w

## PMWIN.DLL Trace Events

The tracepoints for the PMWIN.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for PMWIN Major Code: 0X00C2, sorted by minor code.  
Trace events for PMWIN Major Code: 0X00C2 ,sorted by tracepoint.

## Trace Events for PMWIN Major Code: 0X00C2, Sorted by Minor Code

```

00100 (0X0064) WINREGISTERCLASS PRE-INVOCATION
00101 (0X0065) WINDEFWINDOWPROC PRE-INVOCATION
00102 (0X0066) WINDESTROYWINDOW PRE-INVOCATION
00103 (0X0067) WINSHOWWINDOW PRE-INVOCATION
00104 (0X0068) WINQUERYWINDOWRECT PRE-INVOCATION
00105 (0X0069) WINCREATEWINDOW PRE-INVOCATION
00106 (0X006A) WINCREATESTDWINDOW PRE-INVOCATION
00107 (0X006B) WINENABLEWINDOW PRE-INVOCATION
00108 (0X006C) WINISWINDOWENABLED PRE-INVOCATION
00109 (0X006D) WINISWINDOWVISIBLE PRE-INVOCATION
00110 (0X006E) WINQUERYWINDOWTEXT PRE-INVOCATION
00111 (0X006F) WINSETWINDOWTEXT PRE-INVOCATION
00112 (0X0070) WINQUERYWINDOWTEXTLENGTH PRE-INVOCATION
00113 (0X0071) WINWINDOWFROMID PRE-INVOCATION
00114 (0X0072) WINMULTWINDOWFROMIDS PRE-INVOCATION
00115 (0X0073) WINISWINDOW PRE-INVOCATION
00116 (0X0074) WINQUERYWINDOW PRE-INVOCATION
00117 (0X0075) WINISCHILD PRE-INVOCATION
00118 (0X0076) WINQUERYWINDOWPROCESS PRE-INVOCATION
00119 (0X0077) WINQUERYOBJECTWINDOW PRE-INVOCATION

```

00120 (0X0078) WINQUERYDESKTOPWINDOW PRE-INVOCATION  
00121 (0X0079) WINSUBCLASSWINDOW PRE-INVOCATION  
00122 (0X007A) WINQUERYCLASSNAME PRE-INVOCATION  
00123 (0X007B) WINQUERYCLASSINFO PRE-INVOCATION  
00124 (0X007C) WINQUERYACTIVEWINDOW PRE-INVOCATION  
00125 (0X007D) WINQUERYSYSMODALWINDOW PRE-INVOCATION  
00126 (0X007E) WINSETSYSMODALWINDOW PRE-INVOCATION  
00127 (0X007F) WINLOCKWINDOW PRE-INVOCATION  
00128 (0X0080) WINREGISTERWINDOWDESTROY PRE-INVOCATION  
00129 (0X0081) WINQUERYWINDOWLOCKCOUNT PRE-INVOCATION  
00130 (0X0082) WINQUERYWINDOWUSHORT PRE-INVOCATION  
00131 (0X0083) WINSETWINDOWUSHORT PRE-INVOCATION  
00132 (0X0084) WINQUERYWINDOWULONG/WINQUERYWINDOWPTR PRE-INVOCATION  
00133 (0X0085) WINSETWINDOWULONG/WINSETWINDOWPTR PRE-INVOCATION  
00134 (0X0086) WINSETWINDOWBITS PRE-INVOCATION  
00135 (0X0087) WINCANCELSHUTDOWN PRE-INVOCATION  
00150 (0X0096) WINLISTBOXWNDPROC PRE-INVOCATION  
00151 (0X0097) WINDEFQUEUEPROC PRE-INVOCATION  
00152 (0X0098) WINDESKTOPWNDPROC PRE-INVOCATION  
00153 (0X0099) WINQUEUEWNDPROC PRE-INVOCATION  
00154 (0X009A) WINSETQUEUEPROC PRE-INVOCATION  
00156 (0X009C) WINDOWUPDATERECT PRE-INVOCATION  
00157 (0X009D) WINSAVEWINDOWPOS PRE-INVOCATION  
00158 (0X009E) WINSYSTEMSHUTDOWN PRE-INVOCATION  
00162 (0X00A2) COPYWINDOWRECT PRE-INVOCATION  
00163 (0X00A3) FINDTOPWINDOW PRE-INVOCATION  
00200 (0X00C8) WINREGISTERCLASS POST-INVOCATION  
00201 (0X00C9) WINDEFWINDOWPROC POST-INVOCATION  
00202 (0X00CA) WINDESTROYWINDOW POST-INVOCATION  
00203 (0X00CB) WINSHOWWINDOW POST-INVOCATION  
00204 (0X00CC) WINQUERYWINDOWRECT POST-INVOCATION  
00205 (0X00CD) WINBEGINPAINT POST-INVOCATION  
00206 (0X00CE) WINENDPAINT POST-INVOCATION  
00207 (0X00CF) WINGETPS POST-INVOCATION  
00208 (0X00D0) WINGETCLIPPS POST-INVOCATION  
00209 (0X00D1) WINRELEASEPS POST-INVOCATION  
00210 (0X00D2) WINOPENWINDOWDC POST-INVOCATION  
00211 (0X00D3) WINSCROLLWINDOW POST-INVOCATION  
00212 (0X00D4) WINFILLRECT POST-INVOCATION  
00213 (0X00D5) WINCREATEWINDOW POST-INVOCATION  
00214 (0X00D6) WINCREATESTDWINDOW POST-INVOCATION  
00215 (0X00D7) WINENABLEWINDOW POST-INVOCATION  
00216 (0X00D8) WINISWINDOWENABLED POST-INVOCATION  
00217 (0X00D9) WINENABLEWINDOWUPDATE POST-INVOCATION  
00218 (0X00DA) WINISWINDOWVISIBLE POST-INVOCATION  
00219 (0X00DB) WINQUERYWINDOWTEXT POST-INVOCATION  
00220 (0X00DC) WINSETWINDOWTEXT POST-INVOCATION  
00221 (0X00DD) WINQUERYWINDOWTEXTLENGTH POST-INVOCATION  
00222 (0X00DE) WINWINDOWFROMID POST-INVOCATION  
00223 (0X00DF) WINMULTWINDOWFROMIDS POST-INVOCATION  
00224 (0X00E0) WINISWINDOW POST-INVOCATION  
00225 (0X00E1) WINQUERYWINDOW POST-INVOCATION  
00226 (0X00E2) WINSETPARENT POST-INVOCATION  
00227 (0X00E3) WINISCHILD POST-INVOCATION  
00228 (0X00E4) WINSETOWNER POST-INVOCATION  
00229 (0X00E5) WINQUERYWINDOWPROCESS POST-INVOCATION  
00230 (0X00E6) WINQUERYOBJECTWINDOW POST-INVOCATION  
00231 (0X00E7) WINQUERYDESKTOPWINDOW POST-INVOCATION  
00232 (0X00E8) WINLOADSTRING POST-INVOCATION  
00233 (0X00E9) WINLOADMESSAGE POST-INVOCATION  
00234 (0X00EA) WINQUERYVERSION POST-INVOCATION  
00235 (0X00EB) WININITIALIZE POST-INVOCATION  
00236 (0X00EC) WINTERMINATE POST-INVOCATION  
00300 (0X012C) WINSETWINDOWPOS PRE-INVOCATION  
00301 (0X012D) WINQUERYWINDOWPOS PRE-INVOCATION  
00302 (0X012E) WINSETMULTWINDOWPOS PRE-INVOCATION  
00303 (0X012F) WINSETPARENT PRE-INVOCATION  
00304 (0X0130) WINSETOWNER PRE-INVOCATION  
00305 (0X0131) WINUPDATEWINDOW PRE-INVOCATION  
00306 (0X0132) WININVALIDATERECT PRE-INVOCATION  
00307 (0X0133) WININVALIDATEREGION PRE-INVOCATION  
00308 (0X0134) WINDRAWTEXT PRE-INVOCATION  
00309 (0X0135) WININVALIDATERECT PRE-INVOCATION



00310 (0X0136) WINVALIDATEREGION PRE-INVOCATION  
00311 (0X0137) WINWINDOWFROMDC PRE-INVOCATION  
00312 (0X0138) WINQUERYWINDOWDC PRE-INVOCATION  
00313 (0X0139) WINGETSCREENPS PRE-INVOCATION  
00314 (0X013A) WINQUERYUPDATERECT PRE-INVOCATION  
00315 (0X013B) WINQUERYUPDATEREGION PRE-INVOCATION  
00316 (0X013C) WINEXCLUDEUPDATEREGION PRE-INVOCATION  
00317 (0X013D) WINLOCKWINDOWUPDATE PRE-INVOCATION  
00318 (0X013E) WINLOCKVISREGIONS PRE-INVOCATION  
00319 (0X013F) WINBEGINPAINT PRE-INVOCATION  
00320 (0X0140) WINENDPAINT PRE-INVOCATION  
00321 (0X0141) WINGETPS PRE-INVOCATION  
00322 (0X0142) WINGETCLIPPS PRE-INVOCATION  
00323 (0X0143) WINRELEASEPS PRE-INVOCATION  
00324 (0X0144) WINOPENWINDOWDC PRE-INVOCATION  
00325 (0X0145) WINSCROLLWINDOW PRE-INVOCATION  
00326 (0X0146) WINFILLRECT PRE-INVOCATION  
00327 (0X0147) WINENABLEWINDOWUPDATE PRE-INVOCATION  
00400 (0X0190) WINSETWINDOWPOS POST-INVOCATION  
00401 (0X0191) WINQUERYWINDOWPOS POST-INVOCATION  
00402 (0X0192) WINSETMULTWINDOWPOS POST-INVOCATION  
00403 (0X0193) WINUPDATEWINDOW POST-INVOCATION  
00404 (0X0194) WININVALIDATERECT POST-INVOCATION  
00405 (0X0195) WININVALIDATEREGION POST-INVOCATION  
00406 (0X0196) WININVERTRECT POST-INVOCATION  
00407 (0X0197) WINDRAWBITMAP POST-INVOCATION  
00408 (0X0198) WINDRAWTEXT POST-INVOCATION  
00409 (0X0199) WINDRAWBORDER POST-INVOCATION  
00410 (0X019A) WININVALIDATERECT POST-INVOCATION  
00411 (0X019B) WININVALIDATEREGION POST-INVOCATION  
00412 (0X019C) WINWINDOWFROMDC POST-INVOCATION  
00413 (0X019D) WINQUERYWINDOWDC POST-INVOCATION  
00414 (0X019E) WINGETSCREENPS POST-INVOCATION  
00415 (0X019F) WINQUERYUPDATERECT POST-INVOCATION  
00416 (0X01A0) WINQUERYUPDATEREGION POST-INVOCATION  
00417 (0X01A1) WINEXCLUDEUPDATEREGION POST-INVOCATION  
00418 (0X01A2) WINLOCKWINDOWUPDATE POST-INVOCATION  
00419 (0X01A3) WINLOCKVISREGIONS POST-INVOCATION  
00500 (0X01F4) WINBEGINENUMWINDOWS PRE-INVOCATION  
00501 (0X01F5) WINGETNEXTWINDOW PRE-INVOCATION  
00502 (0X01F6) WINENDENUMWINDOWS PRE-INVOCATION  
00503 (0X01F7) WINWINDOWFROMPOINT PRE-INVOCATION  
00504 (0X01F8) WINMAPWINDOWPOINTS PRE-INVOCATION  
00600 (0X0258) WINSETACTIVEWINDOW POST-INVOCATION  
00601 (0X0259) WINSUBCLASSWINDOW POST-INVOCATION  
00602 (0X025A) WINQUERYCLASSNAME POST-INVOCATION  
00603 (0X025B) WINQUERYCLASSINFO POST-INVOCATION  
00604 (0X025C) WINQUERYACTIVEWINDOW POST-INVOCATION  
00605 (0X025D) WINTHREADACTIVE POST-INVOCATION  
00606 (0X025E) WINQUERYSYSMODALWINDOW POST-INVOCATION  
00607 (0X025F) WINSETSYSMODALWINDOW POST-INVOCATION  
00608 (0X0260) WINLOCKWINDOW POST-INVOCATION  
00609 (0X0261) WINREGISTERWINDOWDESTROY POST-INVOCATION  
00610 (0X0262) WINQUERYWINDOWLOCKCOUNT POST-INVOCATION  
00611 (0X0263) WINQUERYWINDOWUSHORT POST-INVOCATION  
00612 (0X0264) WINSETWINDOWUSHORT POST-INVOCATION  
00613 (0X0265) WINQUERYWINDOWUSHORT/WINQUERYWINDOWULONG/WINQUERYWINDOWPTR  
POST-INVOCATION  
00614 (0X0266) WINSETWINDOWUSHORT/WINSETWINDOWULONG/WINSETWINDOWPTR POST-INVOCATION  
00615 (0X0267) WINSETWINDOWBITS POST-INVOCATION  
00650 (0X028A) WINBEGINENUMWINDOWS POST-INVOCATION  
00651 (0X028B) WINGETNEXTWINDOW POST-INVOCATION  
00652 (0X028C) WINENDENUMWINDOWS POST-INVOCATION  
00653 (0X028D) WINWINDOWFROMPOINT POST-INVOCATION  
00654 (0X028E) WINMAPWINDOWPOINTS POST-INVOCATION  
00700 (0X02BC) WINQUERYQUEUEINFO PRE-INVOCATION  
00701 (0X02BD) WINCREATMSGQUEUE PRE-INVOCATION  
00702 (0X02BE) WINDESTROYMSGQUEUE PRE-INVOCATION  
00703 (0X02BF) WINGETMSG PRE-INVOCATION  
00704 (0X02C0) WINPEEKMSG PRE-INVOCATION  
00705 (0X02C1) WINDISPATCHMSG PRE-INVOCATION  
00706 (0X02C2) WINPOSTMSG PRE-INVOCATION  
00707 (0X02C3) WININSENDMSG PRE-INVOCATION

00708 (0X02C4) WINBROADCASTMSG PRE-INVOCATION  
00709 (0X02C5) WINWAITMSG PRE-INVOCATION  
00710 (0X02C6) WINQUERYQUEUESTATUS PRE-INVOCATION  
00711 (0X02C7) WINPOSTQUEUEMSG PRE-INVOCATION  
00712 (0X02C8) WINQUERYMSGPOS PRE-INVOCATION  
00713 (0X02C9) WINQUERYMSGTIME PRE-INVOCATION  
00714 (0X02CA) WINMSGSEMWAIT PRE-INVOCATION  
00715 (0X02CB) WINMSGMUXSEMWAIT PRE-INVOCATION  
00717 (0X02CD) WINSETMSGINTEREST PRE-INVOCATION  
00718 (0X02CE) WINSENDMSG PRE-INVOCATION  
00751 (0X02EF) WINSENDQUEUEMSG PRE-INVOCATION  
00752 (0X02F0) CREATEQUEUE PRE-INVOCATION  
00753 (0X02F1) FREEQUEUE PRE-INVOCATION  
00754 (0X02F2) CLEARQUEUEGLOBALS PRE-INVOCATION  
00755 (0X02F3) REASSIGNINPUT PRE-INVOCATION  
00756 (0X02F4) ASSOCIATEQUEUE PRE-INVOCATION  
00757 (0X02F5) READMESSAGE PRE-INVOCATION  
00800 (0X0320) WINQUERYQUEUEINFO POST-INVOCATION  
00801 (0X0321) WINCREATEMSGQUEUE POST-INVOCATION  
00802 (0X0322) WINDESTROYMSGQUEUE POST-INVOCATION  
00803 (0X0323) WINCANCELSHUTDOWN POST-INVOCATION  
00804 (0X0324) WINGETMSG/WINPEEKMSG POST-INVOCATION  
00805 (0X0325) WINPEEKMSG POST-INVOCATION  
00806 (0X0326) WINDISPATCHMSG POST-INVOCATION  
00807 (0X0327) WINPOSTMSG/WINPOSTQUEUEMSG POST-INVOCATION  
00808 (0X0328) WININSENDMSG POST-INVOCATION  
00809 (0X0329) WINBROADCASTMSG POST-INVOCATION  
00810 (0X032A) WINWAITMSG POST-INVOCATION  
00811 (0X032B) WINQUERYQUEUESTATUS POST-INVOCATION  
00812 (0X032C) WINPOSTQUEUEMSG POST-INVOCATION  
00813 (0X032D) WINQUERYMSGPOS POST-INVOCATION  
00814 (0X032E) WINQUERYMSGTIME POST-INVOCATION  
00815 (0X032F) WINMSGSEMWAIT POST-INVOCATION  
00816 (0X0330) WINMSGSEMWAIT/WINMSGMUXSEMWAIT POST-INVOCATION  
00817 (0X0331) WINSETMSGINTEREST POST-INVOCATION  
00818 (0X0332) WINSENDMSG/WINSENDQUEUEMSG POST-INVOCATION  
00900 (0X0384) WINSETFOCUS PRE-INVOCATION  
00901 (0X0385) WINFOCUSCHANGE PRE-INVOCATION  
00902 (0X0386) WINSETCAPTURE PRE-INVOCATION  
00903 (0X0387) WINQUERYCAPTURE PRE-INVOCATION  
00904 (0X0388) WINQUERYFOCUS PRE-INVOCATION  
00905 (0X0389) WINSETACTIVEWINDOW PRE-INVOCATION  
00906 (0X038A) WINISTHREADACTIVE PRE-INVOCATION  
00907 (0X038B) WINGETKEYSTATE PRE-INVOCATION  
00908 (0X038C) WINGETPHYSKEYSTATE PRE-INVOCATION  
00909 (0X038D) WINENABLEPHYSINPUT PRE-INVOCATION  
00910 (0X038E) WINISPHYSINPUTENABLED PRE-INVOCATION  
00911 (0X038F) WINSETKEYBOARDSTATETABLE PRE-INVOCATION  
00912 (0X0390) WINTRACKRECT PRE-INVOCATION  
00913 (0X0391) WINSHOWTRACKRECT PRE-INVOCATION  
01000 (0X03E8) WINSETFOCUS POST-INVOCATION  
01001 (0X03E9) WINFOCUSCHANGE POST-INVOCATION  
01002 (0X03EA) WINSETCAPTURE POST-INVOCATION  
01003 (0X03EB) WINQUERYCAPTURE POST-INVOCATION  
01004 (0X03EC) WINQUERYFOCUS/WINQUERYSYSMODALWINDOW POST-INVOCATION  
01005 (0X03ED) WINGETKEYSTATE POST-INVOCATION  
01006 (0X03EE) WINGETPHYSKEYSTATE POST-INVOCATION  
01007 (0X03EF) WINENABLEPHYSINPUT POST-INVOCATION  
01008 (0X03F0) WINISPHYSINPUTENABLED POST-INVOCATION  
01009 (0X03F1) WINSETKEYBOARDSTATETABLE POST-INVOCATION  
01100 (0X044C) WINLOADDLG PRE-INVOCATION  
01101 (0X044D) WINDLGBOX PRE-INVOCATION  
01102 (0X044E) WINDISMISSDLG PRE-INVOCATION  
01103 (0X044F) WINSETDLGITEMSHORT PRE-INVOCATION  
01104 (0X0450) WINQUERYDLGITEMSHORT PRE-INVOCATION  
01105 (0X0451) WINSETDLGITEMTEXT PRE-INVOCATION  
01106 (0X0452) WINQUERYDLGITEMTEXT PRE-INVOCATION  
01107 (0X0453) WINDEFDLGPROC PRE-INVOCATION  
01108 (0X0454) WINALARM PRE-INVOCATION  
01109 (0X0455) WINMESSAGEBOX PRE-INVOCATION  
01110 (0X0456) WINPROCESSDLG PRE-INVOCATION  
01111 (0X0457) WINSENDLGITEMMSG PRE-INVOCATION  
01112 (0X0458) WINMAPDLGPOINTS PRE-INVOCATION

01113 (0X0459) WINSUBSTITUTESTRINGS PRE-INVOCATION  
01114 (0X045A) WINENUMDLGITEM PRE-INVOCATION  
01115 (0X045B) WINCREATEDLG PRE-INVOCATION  
01150 (0X047E) WINEDITWNDPROC PRE-INVOCATION  
01200 (0X04B0) WINLOADDLG POST-INVOCATION  
01201 (0X04B1) WINDLGBOX POST-INVOCATION  
01202 (0X04B2) WINDISMISSDLG POST-INVOCATION  
01203 (0X04B3) WINSETDLGITEMSHORT POST-INVOCATION  
01204 (0X04B4) WINQUERYDLGITEMSHORT POST-INVOCATION  
01205 (0X04B5) WINSETDLGITEMTEXT POST-INVOCATION  
01206 (0X04B6) WINQUERYDLGITEMTEXT POST-INVOCATION  
01207 (0X04B7) WINDEFDLGPROC POST-INVOCATION  
01208 (0X04B8) WINALARM POST-INVOCATION  
01209 (0X04B9) WINMESSAGEBOX POST-INVOCATION  
01210 (0X04BA) WINPROCESSDLG POST-INVOCATION  
01211 (0X04BB) WINSENDDLGITEMMSG POST-INVOCATION  
01212 (0X04BC) WINMAPDLGPOINTS POST-INVOCATION  
01213 (0X04BD) WINSUBSTITUTESTRINGS POST-INVOCATION  
01214 (0X04BE) WINENUMDLGITEM POST-INVOCATION  
01215 (0X04BF) WINCREATEDLG POST-INVOCATION  
01300 (0X0514) WINLOADMENU PRE-INVOCATION  
01301 (0X0515) WINCREATEMENU PRE-INVOCATION  
01302 (0X0516) WINFLASHWINDOW PRE-INVOCATION  
01303 (0X0517) WINCREATEFRAMECONTROLS PRE-INVOCATION  
01304 (0X0518) WINFORMATFRAME PRE-INVOCATION  
01305 (0X0519) WINCALCFRAMERECT PRE-INVOCATION  
01306 (0X051A) WINDRAWBITMAP PRE-INVOCATION  
01307 (0X051B) WINDRAWBORDER PRE-INVOCATION  
01308 (0X051C) WINGETMINPOSITION PRE-INVOCATION  
01309 (0X051D) WINGETMAXPOSITION PRE-INVOCATION  
01350 (0X0546) WINFRAMEWNDPROC PRE-INVOCATION  
01351 (0X0547) WINSCROLLBARWNDPROC PRE-INVOCATION  
01352 (0X0548) WINTITLEBARWNDPROC PRE-INVOCATION  
01353 (0X0549) WINSTATICWNDPROC PRE-INVOCATION  
01354 (0X054A) WINCONTEXTWNDPROC PRE-INVOCATION  
01355 (0X054B) WINCALLHELPHOOK PRE-INVOCATION  
01356 (0X054C) DESTROYLIST PRE-INVOCATION  
01357 (0X054D) INSERTLISTITEM PRE-INVOCATION  
01358 (0X054E) DELETETITLEITEM PRE-INVOCATION  
01359 (0X054F) GETLISTITEMLENGTH PRE-INVOCATION  
01360 (0X0550) GETLISTITEM PRE-INVOCATION  
01362 (0X0552) SETLISTITEM PRE-INVOCATION  
01400 (0X0578) WINLOADMENU POST-INVOCATION  
01401 (0X0579) WINCREATEMENU POST-INVOCATION  
01402 (0X057A) WINFLASHWINDOW POST-INVOCATION  
01403 (0X057B) WINCREATEFRAMECONTROLS POST-INVOCATION  
01404 (0X057C) WINFORMATFRAME POST-INVOCATION  
01405 (0X057D) WINCALCFRAMERECT POST-INVOCATION  
01406 (0X057E) WINGETMINPOSITION POST-INVOCATION  
01407 (0X057F) WINGETMAXPOSITION POST-INVOCATION  
01500 (0X05DC) WINSETRECT PRE-INVOCATION  
01501 (0X05DD) WINISRECTEMPTY PRE-INVOCATION  
01502 (0X05DE) WINCOPYRECT PRE-INVOCATION  
01503 (0X05DF) WINEQUALRECT PRE-INVOCATION  
01504 (0X05E0) WINSETRECTEMPTY PRE-INVOCATION  
01505 (0X05E1) WINOFFSETRECT PRE-INVOCATION  
01506 (0X05E2) WININFLATERECT PRE-INVOCATION  
01507 (0X05E3) WINPTINRECT PRE-INVOCATION  
01508 (0X05E4) WININTERSECTRECT PRE-INVOCATION  
01509 (0X05E5) WINUNIONRECT PRE-INVOCATION  
01510 (0X05E6) WINSUBTRACTRECT PRE-INVOCATION  
01511 (0X05E7) WININVERTRECT PRE-INVOCATION  
01512 (0X05E8) WINMAKERECT PRE-INVOCATION  
01513 (0X05E9) WINMAKEPOINTS PRE-INVOCATION  
01600 (0X0640) WINSETRECT POST-INVOCATION  
01601 (0X0641) WINISRECTEMPTY POST-INVOCATION  
01602 (0X0642) WINCOPYRECT POST-INVOCATION  
01603 (0X0643) WINEQUALRECT POST-INVOCATION  
01604 (0X0644) WINSETRECTEMPTY POST-INVOCATION  
01605 (0X0645) WINOFFSETRECT POST-INVOCATION  
01606 (0X0646) WININFLATERECT POST-INVOCATION  
01607 (0X0647) WINPTINRECT POST-INVOCATION  
01608 (0X0648) WININTERSECTRECT POST-INVOCATION

01609 (0X0649) WINUNIONRECT POST-INVOCATION  
01610 (0X064A) WINSUBTRACTRECT POST-INVOCATION  
01611 (0X064B) WINMAKERECT POST-INVOCATION  
01612 (0X064C) WINMAKEPOINTS POST-INVOCATION  
01613 (0X064D) WINTRACKRECT POST-INVOCATION  
01614 (0X064E) WINSHOWTRACKRECT POST-INVOCATION  
01700 (0X06A4) WINQUERYVERSION PRE-INVOCATION  
01701 (0X06A5) WININITIALIZE PRE-INVOCATION  
01702 (0X06A6) WINTERMINATE PRE-INVOCATION  
01703 (0X06A7) WINQUERYSYSTEMATOMTABLE PRE-INVOCATION  
01704 (0X06A8) WINQUERYSYSVALUE PRE-INVOCATION  
01705 (0X06A9) WINSETSYSVALUE PRE-INVOCATION  
01706 (0X06AA) WINQUERYSYSCOLOR PRE-INVOCATION  
01707 (0X06AB) WINSETSYSCOLORS PRE-INVOCATION  
01708 (0X06AC) WINCATCH PRE-INVOCATION  
01709 (0X06AD) WINTHROW PRE-INVOCATION  
01710 (0X06AE) WINGETLASTERROR PRE-INVOCATION  
01711 (0X06AF) WINGETERRORINFO PRE-INVOCATION  
01712 (0X06B0) WINFREEERRORINFO PRE-INVOCATION  
01713 (0X06B1) WINSTARTTIMER PRE-INVOCATION  
01714 (0X06B2) WINSTOPTIMER PRE-INVOCATION  
01715 (0X06B3) WINGETCURRENTTIME PRE-INVOCATION  
01750 (0X06D6) WINSETERRORINFO PRE-INVOCATION  
01751 (0X06D7) WINSETLASTERROR PRE-INVOCATION  
01800 (0X0708) WINQUERYSYSVALUE POST-INVOCATION  
01801 (0X0709) WINSETSYSVALUE POST-INVOCATION  
01802 (0X070A) WINQUERYSYSCOLOR POST-INVOCATION  
01803 (0X070B) WINSETSYSCOLORS POST-INVOCATION  
01804 (0X070C) WINCATCH POST-INVOCATION  
01805 (0X070D) WINTHROW POST-INVOCATION  
01806 (0X070E) WINGETLASTERROR POST-INVOCATION  
01807 (0X070F) WINGETERRORINFO POST-INVOCATION  
01808 (0X0710) WINFREEERRORINFO POST-INVOCATION  
01809 (0X0711) WINSTARTTIMER POST-INVOCATION  
01810 (0X0712) WINSTOPTIMER POST-INVOCATION  
01811 (0X0713) WINGETCURRENTTIME POST-INVOCATION  
01900 (0X076C) WINLOADACCELTABLE PRE-INVOCATION  
01901 (0X076D) WINCREATEACCELTABLE PRE-INVOCATION  
01902 (0X076E) WINDESTROYACCELTABLE PRE-INVOCATION  
01903 (0X076F) WINCOPYACCELTABLE PRE-INVOCATION  
01904 (0X0770) WINTRANSLATEACCEL PRE-INVOCATION  
01905 (0X0771) WINSETACCELTABLE PRE-INVOCATION  
01906 (0X0772) WINQUERYACCELTABLE PRE-INVOCATION  
02000 (0X07D0) WINLOADACCELTABLE POST-INVOCATION  
02001 (0X07D1) WINCREATEACCELTABLE POST-INVOCATION  
02002 (0X07D2) WINDESTROYACCELTABLE POST-INVOCATION  
02003 (0X07D3) WINCOPYACCELTABLE POST-INVOCATION  
02004 (0X07D4) WINTRANSLATEACCEL POST-INVOCATION  
02005 (0X07D5) WINSETACCELTABLE POST-INVOCATION  
02006 (0X07D6) WINQUERYACCELTABLE POST-INVOCATION  
02100 (0X0834) WINOPENCLIPBRD PRE-INVOCATION  
02101 (0X0835) WINCLOSECLIPBRD PRE-INVOCATION  
02102 (0X0836) WINEMPTYCLIPBRD PRE-INVOCATION  
02103 (0X0837) WINSETCLIPBRDOWNER PRE-INVOCATION  
02104 (0X0838) WINQUERYCLIPBRDOWNER PRE-INVOCATION  
02105 (0X0839) WINSETCLIPBRDDATA PRE-INVOCATION  
02106 (0X083A) WINQUERYCLIPBRDDATA PRE-INVOCATION  
02107 (0X083B) WINENUMCLIPBRDFMTS PRE-INVOCATION  
02108 (0X083C) WINQUERYCLIPBRDFMTINFO PRE-INVOCATION  
02109 (0X083D) WINSETCLIPBRDVIEWER PRE-INVOCATION  
02110 (0X083E) WINQUERYCLIPBRDVIEWER PRE-INVOCATION  
02200 (0X0898) WINOPENCLIPBRD POST-INVOCATION  
02201 (0X0899) WINCLOSECLIPBRD POST-INVOCATION  
02202 (0X089A) WINEMPTYCLIPBRD POST-INVOCATION  
02203 (0X089B) WINSETCLIPBRDOWNER POST-INVOCATION  
02204 (0X089C) WINQUERYCLIPBRDOWNER/WINQUERYCLIPBRDVIEWER/WINQUERYFOCUS/WINQUERYSYSMODALWINDOW POST-INVOCATION  
02205 (0X089D) WINSETCLIPBRDDATA POST-INVOCATION  
02206 (0X089E) WINQUERYCLIPBRDDATA POST-INVOCATION  
02207 (0X089F) WINENUMCLIPBRDFMTS POST-INVOCATION  
02208 (0X08A0) WINQUERYCLIPBRDFMTINFO POST-INVOCATION  
02209 (0X08A1) WINSETCLIPBRDVIEWER POST-INVOCATION

02210 (0X08A2) WINQUERYCLIPBRDVIEWER POST-INVOCATION  
02300 (0X08FC) WINDESTROYCURSOR PRE-INVOCATION  
02301 (0X08FD) WINSHOWCURSOR PRE-INVOCATION  
02302 (0X08FE) WINCREATECURSOR PRE-INVOCATION  
02303 (0X08FF) WINQUERYCURSORINFO PRE-INVOCATION  
02304 (0X0900) WINSETPOINTER PRE-INVOCATION  
02305 (0X0901) WINSHOWPOINTER PRE-INVOCATION  
02306 (0X0902) WINQUERYSYSPINTER PRE-INVOCATION  
02307 (0X0903) WINLOADPOINTER PRE-INVOCATION  
02308 (0X0904) WINCREATEPOINTER PRE-INVOCATION  
02309 (0X0905) WINDESTROYPOINTER PRE-INVOCATION  
02310 (0X0906) WINQUERYPOINTER PRE-INVOCATION  
02311 (0X0907) WINSETPOINTERPOS PRE-INVOCATION  
02312 (0X0908) WINQUERYPOINTERPOS PRE-INVOCATION  
02313 (0X0909) WINQUERYPOINTERINFO PRE-INVOCATION  
02314 (0X090A) WINDRAWPOINTER PRE-INVOCATION  
02315 (0X090B) WINGETSYSBITMAP PRE-INVOCATION  
02400 (0X0960) WINDESTROYCURSOR POST-INVOCATION  
02401 (0X0961) WINSHOWCURSOR POST-INVOCATION  
02402 (0X0962) WINCREATECURSOR POST-INVOCATION  
02403 (0X0963) WINQUERYCURSORINFO POST-INVOCATION  
02404 (0X0964) WINSETPOINTER POST-INVOCATION  
02405 (0X0965) WINSHOWPOINTER POST-INVOCATION  
02406 (0X0966) WINQUERYSYSPINTER POST-INVOCATION  
02407 (0X0967) WINLOADPOINTER POST-INVOCATION  
02408 (0X0968) WINCREATEPOINTER POST-INVOCATION  
02409 (0X0969) WINDESTROYPOINTER POST-INVOCATION  
02410 (0X096A) WINQUERYPOINTER POST-INVOCATION  
02411 (0X096B) WINSETPOINTERPOS POST-INVOCATION  
02412 (0X096C) WINQUERYPOINTERPOS POST-INVOCATION  
02413 (0X096D) WINQUERYPOINTERINFO POST-INVOCATION  
02414 (0X096E) WINDRAWPOINTER POST-INVOCATION  
02415 (0X096F) WINGETSYSBITMAP POST-INVOCATION  
02500 (0X09C4) WINSETHOOK PRE-INVOCATION  
02501 (0X09C5) WINRELEASEHOOK PRE-INVOCATION  
02502 (0X09C6) WINCALLMSGFILTER PRE-INVOCATION  
02552 (0X09F8) FARCALLHOOK PRE-INVOCATION  
02553 (0X09F9) FREEQUEUEWINDOWHOOKS PRE-INVOCATION  
02600 (0X0A28) WINSETHOOK POST-INVOCATION  
02601 (0X0A29) WINRELEASEHOOK POST-INVOCATION  
02602 (0X0A2A) WINCALLMSGFILTER POST-INVOCATION  
02700 (0X0A8C) WINSETCP PRE-INVOCATION  
02701 (0X0A8D) WINQUERYCP PRE-INVOCATION  
02702 (0X0A8E) WINQUERYCPLIST PRE-INVOCATION  
02703 (0X0A8F) WINCPTRANSLATESTRING PRE-INVOCATION  
02704 (0X0A90) WINCPTRANSLATECHAR PRE-INVOCATION  
02705 (0X0A91) WINUPPER PRE-INVOCATION  
02706 (0X0A92) WINUPPERCHAR PRE-INVOCATION  
02707 (0X0A93) WINNEXTCHAR PRE-INVOCATION  
02708 (0X0A94) WINPREVCHAR PRE-INVOCATION  
02709 (0X0A95) WINCOMPARESTRINGS PRE-INVOCATION  
02710 (0X0A96) WINLOADSTRING PRE-INVOCATION  
02711 (0X0A97) WINLOADMESSAGE PRE-INVOCATION  
02750 (0X0ABE) WINLOADCHARXLATETBL PRE-INVOCATION  
02751 (0X0ABF) WINSETCHARXLATETBL PRE-INVOCATION  
02752 (0X0AC0) WINQUERYCHARXLATETBL PRE-INVOCATION  
02753 (0X0AC1) WINLOADVKEYGLYPHXLATETBL PRE-INVOCATION  
02754 (0X0AC2) WINSETVKEYGLYPHXLATETBL PRE-INVOCATION  
02755 (0X0AC3) WINQUERYVKEYGLYPHXLATETBL PRE-INVOCATION  
02756 (0X0AC4) WINSETKBDLAYOUT PRE-INVOCATION  
02757 (0X0AC5) WINQUERYKBDLAYOUT PRE-INVOCATION  
02800 (0X0AF0) WINSETCP POST-INVOCATION  
02801 (0X0AF1) WINQUERYCP POST-INVOCATION  
02802 (0X0AF2) WINQUERYCPLIST POST-INVOCATION  
02803 (0X0AF3) WINCPTRANSLATESTRING POST-INVOCATION  
02804 (0X0AF4) WINCPTRANSLATECHAR POST-INVOCATION  
02805 (0X0AF5) WINUPPER POST-INVOCATION  
02806 (0X0AF6) WINUPPERCHAR POST-INVOCATION  
02807 (0X0AF7) WINNEXTCHAR POST-INVOCATION  
02808 (0X0AF8) WINPREVCHAR POST-INVOCATION  
02809 (0X0AF9) WINCOMPARESTRINGS POST-INVOCATION  
02900 (0X0B54) WINCREATEHEAP PRE-INVOCATION  
02901 (0X0B55) WINDESTROYHEAP PRE-INVOCATION

02902 (0X0B56) WINAVAILMEM PRE-INVOCATION  
02903 (0X0B57) WINALLOCMEM PRE-INVOCATION  
02904 (0X0B58) WINREALLOCMEM PRE-INVOCATION  
02905 (0X0B59) WINFREEMEM PRE-INVOCATION  
02906 (0X0B5A) WINLOCKHEAP PRE-INVOCATION  
02950 (0X0B86) COMPACTMOVEABLEHEAP PRE-INVOCATION  
02951 (0X0B87) FINDFREEBLOCK PRE-INVOCATION  
02952 (0X0B88) FINDMAXFREEBLOCK PRE-INVOCATION  
02953 (0X0B89) INSERTFREEBLOCK PRE-INVOCATION  
02954 (0X0B8A) SORTFREELIST PRE-INVOCATION  
02955 (0X0B8B) GETSIZEDS PRE-INVOCATION  
02956 (0X0B8C) VALIDATEHEAPHANDLE PRE-INVOCATION

-----

## Trace Events for PMWIN Major Code: 0X00C2, Sorted by Tracepoint

ASSOCIATEQUEUE 00756 (0X02F4)  
CLEARQUEUEGLOBALS 00754 (0X02F2)  
COMPACTMOVEABLEHEAP 02950 (0X0B86)  
COPYWINDOWRECT 00162 (0X00A2)  
CREATEQUEUE 00752 (0X02F0)  
DELETELISTITEM 01358 (0X054E)  
DESTROYLIST 01356 (0X054C)  
DRAWTEXT 00408 (0X0198)  
FARCALLHOOK 02552 (0X09F8)  
FINDFREEBLOCK 02951 (0X0B87)  
FINDMAXFREEBLOCK 02952 (0X0B88)  
FINDTOPWINDOW 00163 (0X00A3)  
FREEQUEUE 00753 (0X02F1)  
FREEQUEUEWINDOWHOOKS 02553 (0X09F9)  
GETLISTITEM 01360 (0X0550)  
GETLISTITEMLENGTH 01359 (0X054F)  
GETSIZEDS 02955 (0X0B8B)  
INSERTFREEBLOCK 02953 (0X0B89)  
INSERTLISTITEM 01357 (0X054D)  
POSTWINALARM 01208 (0X04B8)  
POSTWINBEGINENUMWINDOWS 00650 (0X028A)  
POSTWINBEGINPAINT 00205 (0X00CD)  
POSTWINBROADCASTMSG 00809 (0X0329)  
POSTWINCATCH 01804 (0X070C)  
POSTWINCOMPARESTRINGS 02809 (0X0AF9)  
POSTWINCOPYACCELTABLE 02003 (0X07D3)  
POSTWINCOPYRECT 01602 (0X0642)  
POSTWINCREATEACCELTABLE 02001 (0X07D1)  
POSTWINCREATECURSOR 02402 (0X0962)  
POSTWINCREATEFRAMECONTROLS 01403 (0X057B)  
POSTWINCREATEPOINTER 02408 (0X0968)  
POSTWINDESTROYACCELTABLE 02002 (0X07D2)  
POSTWINDESTROYCURSOR 02400 (0X0960)  
POSTWINDESTROYPOINTER 02409 (0X0969)  
POSTWINDESTROYWINDOW 00202 (0X00CA)  
POSTWINDISPATCHMSG 00806 (0X0326)  
POSTWINDRAWPOINTER 02414 (0X096E)  
POSTWINENABLEPHYSINPUT 01007 (0X03EF)  
POSTWINENABLEWINDOW 00215 (0X00D7)  
POSTWINENABLEWINDOWUPDATE 00217 (0X00D9)  
POSTWINENDENUMWINDOWS 00652 (0X028C)  
POSTWINENDPAINT 00206 (0X00CE)  
POSTWINENUMDLGITEM 01214 (0X04BE)  
POSTWINEQUALRECT 01603 (0X0643)  
POSTWINEXCLUDEUPDATERECTION 00417 (0X01A1)  
POSTWINFILLRECT 00212 (0X00D4)  
POSTWINFOCUSCHANGE 01001 (0X03E9)  
POSTWINGETCLIPPS 00208 (0X00D0)



POSTWINGETCURRENTTIME 01811 (0X0713)  
POSTWINGETKEYSTATE 01005 (0X03ED)  
POSTWINGETMAXPOSITION 01407 (0X057F)  
POSTWINGETMINPOSITION 01406 (0X057E)  
POSTWINGETMSG 00804 (0X0324)  
POSTWINGETNEXTWINDOW 00651 (0X028B)  
POSTWINGETPHYSKEYSTATE 01006 (0X03EE)  
POSTWINGETPS 00207 (0X00CF)  
POSTWINGETSCREENPS 00414 (0X019E)  
POSTWINGETSYSBITMAP 02415 (0X096F)  
POSTWININFLATERECT 01606 (0X0646)  
POSTWININSENDMSG 00808 (0X0328)  
POSTWININTERSECTRECT 01608 (0X0648)  
POSTWININVERTRECT 00406 (0X0196)  
POSTWINISCHILD 00227 (0X00E3)  
POSTWINISPHYSINPUTENABLED 01008 (0X03F0)  
POSTWINISRECTEMPTY 01601 (0X0641)  
POSTWINISTHREADACTIVE 00605 (0X025D)  
POSTWINISWINDOW 00224 (0X00E0)  
POSTWINISWINDOWENABLED 00216 (0X00D8)  
POSTWINISWINDOWVISIBLE 00218 (0X00DA)  
POSTWINLOADACCELTABLE 02000 (0X07D0)  
POSTWINLOADMENU 01400 (0X0578)  
POSTWINLOADPOINTER 02407 (0X0967)  
POSTWINLOCKWINDOWUPDATE 00418 (0X01A2)  
POSTWINMAKEPOINTS 01612 (0X064C)  
POSTWINMAKERECT 01611 (0X064B)  
POSTWINMAPDLGPOINTS 01212 (0X04BC)  
POSTWINMAPWINDOWPOINTS 00654 (0X028E)  
POSTWINMSGMUXSEMWAIT 00816 (0X0330)  
POSTWINMSGSEMWAIT 00815 (0X032F)  
POSTWINMULTWINDOWFROMIDS 00223 (0X00DF)  
POSTWINOFFSETRECT 01605 (0X0645)  
POSTWINOPENWINDOWDC 00210 (0X00D2)  
POSTWINPEEKMSG 00805 (0X0325)  
POSTWINPOSTMSG 00807 (0X0327)  
POSTWINPOSTQUEUEMSG 00812 (0X032C)  
POSTWINPTINRECT 01607 (0X0647)  
POSTWINQUERYACCELTABLE 02006 (0X07D6)  
POSTWINQUERYACTIVEWINDOW 00604 (0X025C)  
POSTWINQUERYCAPTURE 01003 (0X03EB)  
POSTWINQUERYCLASSNAME 00602 (0X025A)  
POSTWINQUERYCLIPBRDOWNER 02204 (0X089C)  
POSTWINQUERYCLIPBRDVIEWER 02210 (0X08A2)  
POSTWINQUERYCURSORINFO 02403 (0X0963)  
POSTWINQUERYDLGITEMSHORT 01204 (0X04B4)  
POSTWINQUERYFOCUS 01004 (0X03EC)  
POSTWINQUERYMSGPOS 00813 (0X032D)  
POSTWINQUERYMSGTIME 00814 (0X032E)  
POSTWINQUERYOBJECTWINDOW 00230 (0X00E6)  
POSTWINQUERYPOINTER 02410 (0X096A)  
POSTWINQUERYPOINTERINFO 02413 (0X096D)  
POSTWINQUERYPOINTERPOS 02412 (0X096C)  
POSTWINQUERYQUEUEINFO 00800 (0X0320)  
POSTWINQUERYQUEUESTATUS 00811 (0X032B)  
POSTWINQUERYSYSMODALWINDOW 00606 (0X025E)  
POSTWINQUERYSYSPOINTER 02406 (0X0966)  
POSTWINQUERYSYSVALUE 01800 (0X0708)  
POSTWINQUERYUPDATERECT 00415 (0X019F)  
POSTWINQUERYUPDATEREGION 00416 (0X01A0)  
POSTWINQUERYWINDOW 00225 (0X00E1)  
POSTWINQUERYWINDOWDC 00413 (0X019D)  
POSTWINQUERYWINDOWLOCKCOUNT 00610 (0X0262)  
POSTWINQUERYWINDOWPOS 00401 (0X0191)  
POSTWINQUERYWINDOWPROCESS 00229 (0X00E5)  
POSTWINQUERYWINDOWRECT 00204 (0X00CC)  
POSTWINQUERYWINDOWTEXT 00219 (0X00DB)  
POSTWINQUERYWINDOWTEXTLENGTH 00221 (0X00DD)  
POSTWINQUERYWINDOWVALUE 00611 (0X0263)  
POSTWINQUERYWINDOWVALUE 00613 (0X0265)  
POSTWINREGISTERWINDOWDESTROY 00609 (0X0261)  
POSTWINRELEASEPS 00209 (0X00D1)  
POSTWINSENDDLGITEMMSG 01211 (0X04BB)

POSTWINSENDMSG 00818 (0X0332)  
POSTWINSETACCELTABLE 02005 (0X07D5)  
POSTWINSETACTIVIEWINDOW 00600 (0X0258)  
POSTWINSETCAPTURE 01002 (0X03EA)  
POSTWINSETDLGITEMSHORT 01203 (0X04B3)  
POSTWINSETFOCUS 01000 (0X03E8)  
POSTWINSETKEYBOARDSTATETABLE 01009 (0X03F1)  
POSTWINSETOWNER 00228 (0X00E4)  
POSTWINSETPARENT 00226 (0X00E2)  
POSTWINSETPOINTER 02404 (0X0964)  
POSTWINSETPOINTERPOS 02411 (0X096B)  
POSTWINSETRECT 01600 (0X0640)  
POSTWINSETRECTEMPTY 01604 (0X0644)  
POSTWINSETSYSMODALWINDOW 00607 (0X025F)  
POSTWINSETSYSVALUE 01801 (0X0709)  
POSTWINSETWINDOWBITS 00615 (0X0267)  
POSTWINSETWINDOWTEXT 00220 (0X00DC)  
POSTWINSETWINDOWVALUE 00612 (0X0264)  
POSTWINSETWINDOWVALUE 00614 (0X0266)  
POSTWINSHOWCURSOR 02401 (0X0961)  
POSTWINSHOWPOINTER 02405 (0X0965)  
POSTWINSHOWTRACKRECT 01614 (0X064E)  
POSTWINSTARTTIMER 01809 (0X0711)  
POSTWINSTOPTIMER 01810 (0X0712)  
POSTWINSUBCLASSWINDOW 00601 (0X0259)  
POSTWINSUBSTITUTESTRINGS 01213 (0X04BD)  
POSTWINSUBTRACTRECT 01610 (0X064A)  
POSTWINTHROW 01805 (0X070D)  
POSTWINTRANSLATEACCEL 02004 (0X07D4)  
POSTWINUNIONRECT 01609 (0X0649)  
POSTWINUPDATEWINDOW 00403 (0X0193)  
POSTWINWAITMSG 00810 (0X032A)  
POSTWINWINDOWFROMDC 00412 (0X019C)  
POSTWINWINDOWFROMID 00222 (0X00DE)  
POSTWINWINDOWFROMPOINT 00653 (0X028D)  
READMESSAGE 00757 (0X02F5)  
REASSIGNINPUT 00755 (0X02F3)  
SETLISTITEM 01362 (0X0552)  
SORTFREELIST 02954 (0X0B8A)  
VALIDATEHEAPHANDLE 02956 (0X0B8C)  
WINALARM 01108 (0X0454)  
WINALLOCMEM 02903 ( 0X0B57)  
WINAVAILMEM 02902 (0X0B56)  
WINBEGINENUMWINDOWS 00500 (0X01F4)  
WINBEGINPAINT 00319 (0X013F)  
WINBROADCASTMSG 00708 (0X02C4)  
WINCALCFRAMERECT 01305 (0X0519)  
WINCALLHELPHOOK 01355 (0X054B)  
WINCALLMSGFILTER 02502 (0X09C6)  
WINCANCELSHUTDOWN 00135 (0X0087)  
WINCATCH 01708 (0X06AC)  
WINCLOSECLIPBRD 02101 (0X0835)  
WINCOMPARESTRINGS 02709 (0X0A95)  
WINCOPYACCELTABLE 01903 (0X076F)  
WINCOPYRECT 01502 (0X05DE)  
WINCPTRANSLATECHAR 02704 (0X0A90)  
WINCPTRANSLATESTRING 02703 (0X0A8F)  
WINCREATEACCELTABLE 01901 (0X076D)  
WINCREATECURSOR 02302 (0X08FE)  
WINCREATEDLG 01115 (0X045B)  
WINCREATEFRAMECONTROLS 01303 (0X0517)  
WINCREATEHEAP 02900 (0X0B54)  
WINCREATEMENU 01301 (0X0515)  
WINCREATEMSGQUEUE 00701 (0X02BD)  
WINCREATEPOINTER 02308 (0X0904)  
WINCREATESTDWINDOW 00106 (0X006A)  
WINCREATEWINDOW 00105 (0X0069)  
WINDEFDLGPROC 01107 (0X0453)  
WINDEFQUEUEPROC 00151 (0X0097)  
WINDEFWINDOWPROC 00101 (0X0065)  
WINDESKTOPWNDPROC 00152 (0X0098)  
WINDESTROYACCELTABLE 01902 (0X076E)  
WINDESTROYCURSOR 02300 (0X08FC)



WINDESTROYHEAP 02901 (0X0B55)  
WINDESTROYMSGQUEUE 00702 (0X02BE)  
WINDESTROYPOINTER 02309 (0X0905)  
WINDESTROYWINDOW 00102 (0X0066)  
WINDISMISSDLG 01102 (0X044E)  
WINDISPATCHMSG 00705 (0X02C1)  
WINDLGBOX 01101 (0X044D)  
WINDOWUPDATERECT 00156 (0X009C)  
WINDRAWBITMAP 01306 (0X051A)  
WINDRAWBORDER 01307 (0X051B)  
WINDRAWPOINTER 02314 (0X090A)  
WINDRAWTEXT 00308 (0X0134)  
WINEDITWNDPROC 01150 (0X047E)  
WINEMPTYCLIPBRD 02102 (0X0836)  
WINENABLEPHYSINPUT 00909 (0X038D)  
WINENABLEWINDOW 00107 (0X006B)  
WINENABLEWINDOWUPDATE 00327 (0X0147)  
WINENDENUMWINDOWS 00502 (0X01F6)  
WINENDPAINT 00320 (0X0140)  
WINENUMCLIPBRDFMTS 02107 (0X083B)  
WINENUMDLGITEM 01114 (0X045A)  
WINEQUALRECT 01503 (0X05DF)  
WINEXCLUDEUPDATEREGION 00316 (0X013C)  
WINFILLRECT 00326 (0X0146)  
WINFLASHWINDOW 01302 (0X0516)  
WINFOCUSCHANGE 00901 (0X0385)  
WINFORMATFRAME 01304 (0X0518)  
WINFRAMEWNDPROC 01350 (0X0546)  
WINFREEERRORINFO 01712 (0X06B0)  
WINFREEMEM 02905 (0X0B59)  
WINGETCLIPPS 00322 (0X0142)  
WINGETCURRENTTIME 01715 (0X06B3)  
WINGETERRORINFO 01711 (0X06AF)  
WINGETKEYSTATE 00907 (0X038B)  
WINGETLASTERROR 01710 (0X06AE)  
WINGETMAXPOSITION 01309 (0X051D)  
WINGETMINPOSITION 01308 (0X051C)  
WINGETMSG 00703 (0X02BF)  
WINGETNEXTWINDOW 00501 (0X01F5)  
WINGETPHYSKEYSTATE 00908 (0X038C)  
WINGETPS 00321 (0X0141)  
WINGETSCREENPS 00313 (0X0139)  
WINGETSYSBITMAP 02315 (0X090B)  
WINCONTEXTWNDPROC 01354 (0X054A)  
WININFLATERECT 01506 (0X05E2)  
WININITIALIZE 01701 (0X06A5)  
WININSENDMSG 00707 (0X02C3)  
WININTERSECTRECT 01508 (0X05E4)  
WININVALIDATERECT 00306 (0X0132)  
WININVALIDATEREGION 00307 ( 0X0133)  
WININVERTRECT 01511 (0X05E7)  
WINISCHILD 00117 (0X0075)  
WINISPHYSINPUTENABLED 00910 (0X038E)  
WINISRECTEMPTY 01501 (0X05DD)  
WINISTHREADACTIVE 00906 (0X038A)  
WINISWINDOW 00115 (0X0073)  
WINISWINDOWENABLED 00108 (0X006C)  
WINISWINDOWVISIBLE 00109 (0X006D)  
WINLISTBOXWNDPROC 00150 (0X0096)  
WINLOADACCELTABLE 01900 (0X076C)  
WINLOADCHARXLATETBL 02750 (0X0ABE)  
WINLOADDLG 01100 (0X044C)  
WINLOADMENU 01300 (0X0514)  
WINLOADMESSAGE 02711 (0X0A97)  
WINLOADPOINTER 02307 (0X0903)  
WINLOADSTRING 02710 (0X0A96)  
WINLOADVKEYGLYPHLATETBL 02753 (0X0AC1)  
WINLOCKHEAP 02906 (0X0B5A)  
WINLOCKVISREGIONS 00318 (0X013E)  
WINLOCKWINDOW 00127 (0X007F)  
WINLOCKWINDOWUPDATE 00317 (0X013D)  
WINMAKEPOINTS 01513 (0X05E9)  
WINMAKERECT 01512 (0X05E8)

WINMAPDLGPOINTS 01112 (0X0458)  
WINMAPWINDOWPOINTS 00504 (0X01F8)  
WINMESSAGEBOX 01109 (0X0455)  
WINMSGMUXSEMWAIT 00715 (0X02CB)  
WINMSGSEMWAIT 00714 (0X02CA)  
WINMULTWINDOWFROMIDS 00114 (0X0072)  
WINNEXTCHAR 02707 (0X0A93)  
WINOFFSETRECT 01505 (0X05E1)  
WINOPENCLIPBRD 02100 (0X0834)  
WINOPENWINDOWDC 00324 (0X0144)  
WINPEEKMSG 00704 (0X02C0)  
WINPOSTMSG 00706 (0X02C2)  
WINPOSTQUEUEMSG 00711 (0X02C7)  
WINPREVCHAR 02708 (0X0A94)  
WINPROCESSDLG 01110 (0X0456)  
WINPTINRECT 01507 (0X05E3)  
WINQUERYACCELTABLE 01906 (0X0772)  
WINQUERYACTIVEWINDOW 00124 (0X007C)  
WINQUERYCAPTURE 00903 (0X0387)  
WINQUERYCHARXLATETBL 02752 (0X0AC0)  
WINQUERYCLASSINFO 00123 (0X007B)  
WINQUERYCLASSNAME 00122 (0X007A)  
WINQUERYCLIPBRDDATA 02106 (0X083A)  
WINQUERYCLIPBRDFMTINFO 02108 (0X083C)  
WINQUERYCLIPBRDOWNER 02104 (0X0838)  
WINQUERYCLIPBRDVIEWER 02110 (0X083E)  
WINQUERYCP 02701 (0X0A8D)  
WINQUERYCPLIST 02702 (0X0A8E)  
WINQUERYCURSORINFO 02303 (0X08FF)  
WINQUERYDESKTOPWINDOW 00120 (0X0078)  
WINQUERYDLGITEMSHORT 01104 (0X0450)  
WINQUERYDLGITEMTEXT 01106 (0X0452)  
WINQUERYFOCUS 00904 (0X0388)  
WINQUERYKBDLAYOUT 02757 (0X0AC5)  
WINQUERYMSGPOS 00712 (0X02C8)  
WINQUERYMSGTIME 00713 (0X02C9)  
WINQUERYOBJECTWINDOW 00119 (0X0077)  
WINQUERYPOINTER 02310 (0X0906)  
WINQUERYPOINTERINFO 02313 (0X0909)  
WINQUERYPOINTERPOS 02312 (0X0908)  
WINQUERYQUEUEINFO 00700 (0X02BC)  
WINQUERYQUEUESTATUS 00710 (0X02C6)  
WINQUERYSYSCOLOR 01706 (0X06AA)  
WINQUERYSYSMODALWINDOW 00125 (0X007D)  
WINQUERYSYSPOINTER 02306 (0X0902)  
WINQUERYSYSTEMATOMTABLE 01703 (0X06A7)  
WINQUERYSYSVALUE 01704 (0X06A8)  
WINQUERYUPDATERECT 00314 (0X013A)  
WINQUERYUPDATEREGION 00315 (0X013B)  
WINQUERYVERSION 01700 (0X06A4)  
WINQUERYVKEYGLYPHXLATETBL 02755 (0X0AC3)  
WINQUERYWINDOW 00116 (0X0074)  
WINQUERYWINDOWDC 00312 (0X0138)  
WINQUERYWINDOWLOCKCOUNT 00129 (0X0081)  
WINQUERYWINDOWPOS 00301 (0X012D)  
WINQUERYWINDOWPROCESS 00118 (0X0076)  
WINQUERYWINDOWRECT 00104 (0X0068)  
WINQUERYWINDOWTEXT 00110 ( 0X006E)  
WINQUERYWINDOWTEXTLENGTH 00112 (0X0070)  
WINQUERYWINDOWULONG 00132 (0X0084)  
WINQUERYWINDOWUSHORT 00130 (0X0082)  
WINQUEUEWNDPROC 00153 (0X0099)  
WINREALLOCMEM 02904 (0X0B58)  
WINREGISTERCLASS 00100 (0X0064)  
WINREGISTERWINDOWDESTROY 00128 (0X0080)  
WINRELEASEHOOK 02501 (0X09C5)  
WINRELEASEPS 00323 (0X0143)  
WINSAVEWINDOWPOS 00157 (0X009D)  
WINSROLLBARWNDPROC 01351 (0X0547)  
WINSROLLWINDOW 00325 (0X0145)  
WINSENDDLGITEMMSG 01111 (0X0457)  
WINSENDMSG 00718 (0X02CE)  
WINSENDQUEUEMSG 00751 (0X02EF)

WINSETACCELTABLE 01905 (0X0771)  
WINSETACTIVIEWINDOW 00905 (0X0389)  
WINSETCAPTURE 00902 (0X0386)  
WINSETCHARXLATETBL 02751 (0X0ABF)  
WINSETCLIPBRDDATA 02105 (0X0839)  
WINSETCLIPBRDOWNER 02103 (0X0837)  
WINSETCLIPBRDVIEWER 02109 (0X083D)  
WINSETCP 02700 (0X0A8C)  
WINSETDLGITEMSHORT 01103 (0X044F)  
WINSETDLGITEMTEXT 01105 (0X0451)  
WINSETFOCUS 00900 (0X0384)  
WINSETHOOK 02500 (0X09C4)  
WINSETKBDLAYOUT 02756 (0X0AC4)  
WINSETKEYBOARDSTATETABLE 00911 (0X038F)  
WINSETLASTERROR 01751 (0X06D7)  
WINSETMSGINTEREST 00717 (0X02CD)  
WINSETMULTWINDOWPOS 00302 (0X012E)  
WINSETOWNER 00304 (0X0130)  
WINSETPARENT 00303 (0X012F)  
WINSETPOINTER 02304 (0X0900)  
WINSETPOINTERPOS 02311 (0X0907)  
WINSETQUEUEPROC 00154 (0X009A)  
WINSETRECT 01500 (0X05DC)  
WINSETRECTEMPTY 01504 (0X05E0)  
WINSETSYS\_COLORS 01707 (0X06AB)  
WINSETSYS\_MODALWINDOW 00126 (0X007E)  
WINSETSYSVALUE 01705 (0X06A9)  
WINSETVKEYGLYPHXLATETBL 02754 (0X0AC2)  
WINSETWINDOWBITS 00134 (0X0086)  
WINSETWINDOWPOS 00300 (0X012C)  
WINSETWINDOWTEXT 00111 (0X006F)  
WINSETWINDOWULONG 00133 (0X0085)  
WINSETWINDOWUSHORT 00131 (0X0083)  
WINSHOWCURSOR 02301 (0X08FD)  
WINSHOWPOINTER 02305 (0X0901)  
WINSHOWTRACKRECT 00913 (0X0391)  
WINSHOWWINDOW 00103 (0X0067)  
WINSTARTTIMER 01713 (0X06B1)  
WINSTATICWNDPROC 01353 (0X0549)  
WINSTOPTIMER 01714 (0X06B2)  
WINSUBCLASSWINDOW 00121 (0X0079)  
WINSUBSTITUTE\_STRINGS 01113 (0X0459)  
WINSUBTRACTRECT 01510 (0X05E6)  
WINSYSTEMSHUTDOWN 00158 (0X009E)  
WINTERMINATE 01702 (0X06A6)  
WINTHROW 01709 (0X06AD)  
WINTITLEBARWNDPROC 01352 (0X0548)  
WINTRACKRECT 00912 (0X0390)  
WINTRANSLATEACCEL 01904 (0X0770)  
WINUNIONRECT 01509 (0X05E5)  
WINUPDATEWINDOW 00305 (0X0131)  
WINUPPER 02705 (0X0A91)  
WINUPPERCHAR 02706 (0X0A92)  
WINVALIDATERECT 00309 (0X0135)  
WINVALIDATEREGION 00310 (0X0136)  
WINWAITMSG 00709 (0X02C5)  
WINWINDOWFROMDC 00311 (0X0137)  
WINWINDOWFROMID 00113 (0X0071)  
WINWINDOWFROMPOINT 00503 (0X01F7)  
WinCalcFrameRect 01405 (0X057D)  
WinCallMsgFilter 02602 (0X0A2A)  
WinCancelShutdown 00803 (0X0323)  
WinCloseClipbrd 02201 (0X0899)  
WinCpTranslateChar 02804 (0X0AF4)  
WinCpTranslateString 02803 (0X0AF3)  
WinCreateDlg 01215 (0X04BF)  
WinCreateMenu 01401 (0X0579)  
WinCreateMsgQueue 00801 (0X0321)  
WinCreateStdWindow 00214 (0X00D6)  
WinCreateWindow 00213 (0X00D5)  
WinDefDlgProc 01207 (0X04B7)  
WinDefWindowProc 00201 (0X00C9)  
WinDestroyMsgQueue 00802 (0X0322)

WinDismissDlg 01202 (0X04B2)  
 WinDlgBox 01201 (0X04B1)  
 WinDrawBitmap 00407 (0X0197)  
 WinDrawBorder 00409 (0X0199)  
 WinEmptyClipbrd 02202 (0X089A)  
 WinEnumClipbrdFmts 02207 (0X089F)  
 WinFlashWindow 01402 (0X057A)  
 WinFormatFrame 01404 (0X057C)  
 WinFreeErrorInfo 01808 (0X0710)  
 WinGetErrorInfo 01807 (0X070F)  
 WinGetLastError 01806 (0X070E)  
 WinInitialize 00235 (0X00EB)  
 WinInvalidateRect 00404 (0X0194)  
 WinInvalidateRegion 00405 (0X0195)  
 WinLoadDlg 01200 (0X04B0)  
 WinLoadMessage 00233 (0X00E9)  
 WinLoadString 00232 (0X00E8)  
 WinLockVisRegions 00419 (0X01A3)  
 WinLockWindow 00608 (0X0260)  
 WinMessageBox 01209 (0X04B9)  
 WinNextChar 02807 (0X0AF7)  
 WinOpenClipbrd 02200 (0X0898)  
 WinPrevChar 02808 (0X0AF8)  
 WinProcessDlg 01210 (0X04BA)  
 WinQueryClassInfo 00603 (0X025B)  
 WinQueryClipbrdData 02206 (0X089E)  
 WinQueryClipbrdFmtInfo 02208 (0X08A0)  
 WinQueryCp 02801 (0X0AF1)  
 WinQueryCpList 02802 (0X0AF2)  
 WinQueryDesktopWindow 00231 (0X00E7)  
 WinQueryDlgItemText 01206 (0X04B6)  
 WinQuerySysColor 01802 (0X070A)  
 WinQueryVersion 00234 (0X00EA)  
 WinRegisterClass 00200 (0X00C8)  
 WinReleaseHook 02601 (0X0A29)  
 WinScrollWindow 00211 (0X00D3)  
 WinSetClipbrdData 02205 (0X089D)  
 WinSetClipbrdOwner 02203 (0X089B)  
 WinSetClipbrdViewer 02209 (0X08A1)  
 WinSetCp 02800 (0X0AF0)  
 WinSetDlgItemText 01205 (0X04B5)  
 WinSetHook 02600 (0X0A28)  
 WinSetMsgInterest 00817 (0X0331)  
 WinSetMultWindowPos 00402 (0X0192)  
 WinSetSysColors 01803 (0X070B)  
 WinSetWindowPos 00400 (0X0190)  
 WinShowWindow 00203 (0X00CB)  
 WinTerminate 00236 (0X00EC)  
 WinTrackRect 01613 (0X064D)  
 WinUpper 02805 (0X0AF5)  
 WinUpperChar 02806 (0X0AF6)  
 WinValidateRect 00410 (0X019A)  
 WinValidateRegion 00411 (0X019B)  
 \_WINSETERRORINFO 01750 (0X06D6)

-----

## PMWIN Major Code: 0X00C2 Minor Code: 100 (0X0064)

### Description

WINREGISTERCLASS PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINREGISTERCLASS

### Minor Code

100 (0X0064)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CBWINDOWDATA = %W, FLSTYLE = %D

PFNWNDPROC = %A, PSZCLASSNAME = %A

PSZCLASSNAME -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 101 (0X0065)

**Description**

WINDEFWINDOWPROC PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDEFWINDOWPROC

**Minor Code**

101 (0X0065)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 102 (0X0066)

**Description**

WINDESTROYWINDOW PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYWINDOW

**Minor Code**

102 (0X0066)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWND = %A

---

## PMWIN Major Code: 0X00C2 Minor Code: 103 (0X0067)

**Description**

WINSHOWWINDOW PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSHOWWINDOW

**Minor Code**

103 (0X0067)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FSHOW = %W, HWND = %A

---

## PMWIN Major Code: 0X00C2 Minor Code: 104 (0X0068)

**Description**

WINQUERYWINDOWRECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWRECT

**Minor Code**

104 (0X0068)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RCLDEST = %A, HWND = %A

---

## PMWIN Major Code: 0X00C2 Minor Code: 105 (0X0069)

**Description**

WINCREATEWINDOW PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCREATEWINDOW

**Minor Code**

105 (0X0069)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RESPARAMS = %A, PCTLDATA = %A, ID = %W

HWNDINSERTBEHIND = %A, HWNDOWNER = %A, CY = %W

CX = %W, Y = %W, X = %W, FLSTYLE = %D

PSZNAME = %A, PSZCLASS = %A, HWNDPARENT = %A

%CB = %W, FLCREATEFLAGS = %D

HMODRESOURCES = %W, IDRESOURCES = %W

PSZNAME -> %S

PSZCLASS -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 106 (0X006A)

**Description**

WINCREATESTDWINDOW PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCREATESTDWINDOW

**Minor Code**

106 (0X006A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWNDCLIENT = %A, IDRESOURCES = %W, HMOD = %W

STYLECLIENT = %D, PSZTITLE = %A, PSZCLIENTCLASS = %A

PFLCREATEFLAGS = %A, FLSTYLE = %D, HWNDPARENT = %A

%HWNDCLIENT -> %A

%FLCREATEFLAGS -> %D

PSZCLIENTCLASS -> %S

PSZTITLE -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 107 (0X006B)

<u>Description</u>	WINENABLEWINDOW PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINENABLEWINDOW
<u>Minor Code</u>	107 (0X006B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%FENABLE = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 108 (0X006C)

<u>Description</u>	WINISWINDOWENABLED PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINISWINDOWENABLED
<u>Minor Code</u>	108 (0X006C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 109 (0X006D)

<u>Description</u>	WINISWINDOWVISIBLE PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINISWINDOWVISIBLE



**Minor Code** 109 (0X006D)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 110 (0X006E)

**Description** WINQUERYWINDOWTEXT PRE-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWTEXT

**Minor Code** 110 (0X006E)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

%SZBUFFER = %A, CCHBUFFERMAX = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 111 (0X006F)

**Description** WINSETWINDOWTEXT PRE-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWTEXT

**Minor Code** 111 (0X006F)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

%SZTEXT = %A, HWND = %A

PSZTEXT -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 112 (0X0070)

### Description

WINQUERYWINDOWTEXTLENGTH PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWTEXTLENGTH

### Minor Code

112 (0X0070)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 113 (0X0071)

### Description

WINWINDOWFROMID PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINWINDOWFROMID

### Minor Code

113 (0X0071)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%ID = %W, HWNDPARENT = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 114 (0X0072)

### Description

WINMULTWINDOWFROMIDS PRE-INVOCATION

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINMULTWINDOWFROMIDS
<b><u>Minor Code</u></b>	114 (0X0072)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%IDLAST = %W, IDFIRST = %W PRGHWND = %A, HWNDPARENT = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 115 (0X0073)

<b><u>Description</u></b>	WINISWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINISWINDOW
<b><u>Minor Code</u></b>	115 (0X0073)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 116 (0X0074)

<b><u>Description</u></b>	WINQUERYWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOW
<b><u>Minor Code</u></b>	116 (0X0074)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%FLOCK = %W, CMD = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 117 (0X0075)

**Description**

WINISCHILD PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINISCHILD

**Minor Code**

117 (0X0075)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWNDPARENT = %A, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 118 (0X0076)

**Description**

WINQUERYWINDOWPROCESS PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWPROCESS

**Minor Code**

118 (0X0076)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%TID = %A, PPID = %A, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 119 (0X0077)

<b><u>Description</u></b>	WINQUERYOBJECTWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYOBJECTWINDOW
<b><u>Minor Code</u></b>	119 (0X0077)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWNDDESKTOP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 120 (0X0078)

<b><u>Description</u></b>	WINQUERYDESKTOPWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYDESKTOPWINDOW
<b><u>Minor Code</u></b>	120 (0X0078)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HDC = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 121 (0X0079)

<b><u>Description</u></b>	WINSUBCLASSWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSUBCLASSWINDOW
<b><u>Minor Code</u></b>	121 (0X0079)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%FNWP = %A, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 122 (0X007A)

**Description**

WINQUERYCLASSNAME PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLASSNAME

**Minor Code**

122 (0X007A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CH = %A, CCHMAX = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 123 (0X007B)

**Description**

WINQUERYCLASSINFO PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLASSINFO

**Minor Code**

123 (0X007B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CLASSINFO = %A, PSZCLASSNAME = %A

PSZCLASSNAME -> %S

-----

PMWIN Major Code: 0X00C2 Minor Code: 124 (0X007C)

<b>Description</b>	WINQUERYACTIVEWINDOW PRE-INVOCATION
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYACTIVEWINDOW
<b>Minor Code</b>	124 (0X007C)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	%FLOCK = %W, HWNDDDESKTOP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 125 (0X007D)

<b>Description</b>	WINQUERYSYSMODALWINDOW PRE-INVOCATION
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSMODALWINDOW
<b>Minor Code</b>	125 (0X007D)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	%FLOCK = %W, HWNDDDESKTOP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 126 (0X007E)

<b>Description</b>	WINSETSYSMODALWINDOW PRE-INVOCATION
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETSYSMODALWINDOW
<b>Minor Code</b>	126 (0X007E)
<b>Trace Groups</b>	No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%HWND = %A, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 127 (0X007F)

**Description**  
WINLOCKWINDOW PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINLOCKWINDOW

**Minor Code**  
127 (0X007F)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%FLOCK = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 128 (0X0080)

**Description**  
WINREGISTERWINDOWDESTROY PRE-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WINREGISTERWINDOWDESTROY

**Minor Code**  
128 (0X0080)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%FREGISTER = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 129 (0X0081)



<u>Description</u>	WINQUERYWINDOWLOCKCOUNT PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWLOCKCOUNT
<u>Minor Code</u>	129 (0X0081)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%HWND = %A

## PMWIN Major Code: 0X00C2 Minor Code: 130 (0X0082)

<u>Description</u>	WINQUERYWINDOWUSHORT PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWUSHORT
<u>Minor Code</u>	130 (0X0082)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%INDEX = %W, HWND = %A

## PMWIN Major Code: 0X00C2 Minor Code: 131 (0X0083)

<u>Description</u>	WINSETWINDOWUSHORT PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWUSHORT
<u>Minor Code</u>	131 (0X0083)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%US = %W, INDEX = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 132 (0X0084)

**Description**

WINQUERYWINDOWULONG/WINQUERYWINDOWPTR PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWULONG

**Minor Code**

132 (0X0084)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%INDEX = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 133 (0X0085)

**Description**

WINSETWINDOWULONG/WINSETWINDOWPTR PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWULONG

**Minor Code**

133 (0X0085)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%UL = %D, INDEX = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 134 (0X0086)

<u>Description</u>	WINSETWINDOWBITS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWBITS
<u>Minor Code</u>	134 (0X0086)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%FLMASK = %D, FLDATA = %D, INDEX = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 135 (0X0087)

<u>Description</u>	WINCANCELSHUTDOWN PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINCANCELSHUTDOWN
<u>Minor Code</u>	135 (0X0087)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%FCANCELALWAYS = %W, HMQ = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 150 (0X0096)

<u>Description</u>	WINLISTBOXWNDPROC PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINLISTBOXWNDPROC
<u>Minor Code</u>	150 (0X0096)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 151 (0X0097)

**Description**                      WINDEFQUEUEPROC PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINDEFQUEUEPROC

**Minor Code**                      151 (0X0097)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 152 (0X0098)

**Description**                      WINDESKTOPWNDPROC PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINDESKTOPWNDPROC

**Minor Code**                      152 (0X0098)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 153 (0X0099)

<b><u>Description</u></b>	WINQUEUEWNDPROC PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUEUEWNDPROC
<b><u>Minor Code</u></b>	153 (0X0099)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 154 (0X009A)

<b><u>Description</u></b>	WINSETQUEUEPROC PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETQUEUEPROC
<b><u>Minor Code</u></b>	154 (0X009A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FNQUEUEPROC = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 156 (0X009C)

<b><u>Description</u></b>	WINDOWUPDATERECT PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINDOWUPDATERECT
<b><u>Minor Code</u></b>	156 (0X009C)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

%FSYNCUPDATE = %W, FS = %W, PRCL = %A, PWND = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 157 (0X009D)

**Description**                      WINSAVEWINDOWPOS PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINSAVEWINDOWPOS

**Minor Code**                      157 (0X009D)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

%CSWP = %W, LPSWP = %A, HSVWP = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 158 (0X009E)

**Description**                      WINSYSTEMSHUTDOWN PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINSYSTEMSHUTDOWN

**Minor Code**                      158 (0X009E)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

NO PARAMETERS

-----

PMWIN Major Code: 0X00C2 Minor Code: 162 (0X00A2)

<b><u>Description</u></b>	COPYWINDOWRECT PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.COPYWINDOWRECT
<b><u>Minor Code</u></b>	162 (0X00A2)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%WRC = %A, PWND = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 163 (0X00A3)

<b><u>Description</u></b>	FINDTOPWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.FINDTOPWINDOW
<b><u>Minor Code</u></b>	163 (0X00A3)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%WND = %W, PSWP = %A, CSWP = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 200 (0X00C8)

<b><u>Description</u></b>	WINREGISTERCLASS POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinRegisterClass
<b><u>Minor Code</u></b>	200 (0X00C8)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 201 (0X00C9)

**Description**

WINDEFWINDOWPROC POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinDefWindowProc

**Minor Code**

201 (0X00C9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 202 (0X00CA)

**Description**

WINDESTROYWINDOW POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYWINDOW

**Minor Code**

202 (0X00CA)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 203 (0X00CB)



<u>Description</u>	WINSHOWWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinShowWindow
<u>Minor Code</u>	203 (0X00CB)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 204 (0X00CC)

<u>Description</u>	WINQUERYWINDOWRECT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWRECT
<u>Minor Code</u>	204 (0X00CC)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 205 (0X00CD)

<u>Description</u>	WINBEGINPAINT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINBEGINPAINT
<u>Minor Code</u>	205 (0X00CD)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 206 (0X00CE)

**Description**

WINENDPAINT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINENDPAINT

**Minor Code**

206 (0X00CE)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 207 (0X00CF)

**Description**

WINGETPS POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETPS

**Minor Code**

207 (0X00CF)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 208 (0X00D0)

<u>Description</u>	WINGETCLIPPS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETCLIPPS
<u>Minor Code</u>	208 (0X00D0)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 209 (0X00D1)

<u>Description</u>	WINRELEASEPS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINRELEASEPS
<u>Minor Code</u>	209 (0X00D1)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 210 (0X00D2)

<u>Description</u>	WINOPENWINDOWDC POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINOPENWINDOWDC
<u>Minor Code</u>	210 (0X00D2)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 211 (0X00D3)

**Description**

WINSCROLLWINDOW POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinScrollWindow

**Minor Code**

211 (0X00D3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 212 (0X00D4)

**Description**

WINFILLRECT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINFILLRECT

**Minor Code**

212 (0X00D4)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 213 (0X00D5)

<u>Description</u>	WINCREATEWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinCreateWindow
<u>Minor Code</u>	213 (0X00D5)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 214 (0X00D6)

<u>Description</u>	WINCREATESTDWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinCreateStdWindow
<u>Minor Code</u>	214 (0X00D6)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 215 (0X00D7)

<u>Description</u>	WINENABLEWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINENABLEWINDOW
<u>Minor Code</u>	215 (0X00D7)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 216 (0X00D8)

**Description**

WINISWINDOWENABLED POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINISWINDOWENABLED

**Minor Code**

216 (0X00D8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 217 (0X00D9)

**Description**

WINENABLEWINDOWUPDATE POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINENABLEWINDOWUPDATE

**Minor Code**

217 (0X00D9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 218 (0X00DA)

<u>Description</u>	WINISWINDOWVISIBLE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINISWINDOWVISIBLE
<u>Minor Code</u>	218 (0X00DA)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 219 (0X00DB)

<u>Description</u>	WINQUERYWINDOWTEXT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWTEXT
<u>Minor Code</u>	219 (0X00DB)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 220 (0X00DC)

<u>Description</u>	WINSETWINDOWTEXT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWTEXT
<u>Minor Code</u>	220 (0X00DC)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 221 (0X00DD)

**Description**

WINQUERYWINDOWTEXTLENGTH POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWTEXTLENGTH

**Minor Code**

221 (0X00DD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 222 (0X00DE)

**Description**

WINWINDOWFROMID POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINWINDOWFROMID

**Minor Code**

222 (0X00DE)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 223 (0X00DF)



<b>Description</b>	WINMULTWINDOWFROMIDS POST-INVOCATION
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINMULTWINDOWFROMIDS
<b>Minor Code</b>	223 (0X00DF)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 224 (0X00E0)

<b>Description</b>	WINISWINDOW POST-INVOCATION
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINISWINDOW
<b>Minor Code</b>	224 (0X00E0)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 225 (0X00E1)

<b>Description</b>	WINQUERYWINDOW POST-INVOCATION
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOW
<b>Minor Code</b>	225 (0X00E1)
<b>Trace Groups</b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 226 (0X00E2)

**Description**

WINSETPARENT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETPARENT

**Minor Code**

226 (0X00E2)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 227 (0X00E3)

**Description**

WINISCHILD POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINISCHILD

**Minor Code**

227 (0X00E3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 228 (0X00E4)

<u>Description</u>	WINSETOWNER POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETOWNER
<u>Minor Code</u>	228 (0X00E4)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 229 (0X00E5)

<u>Description</u>	WINQUERYWINDOWPROCESS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWPROCESS
<u>Minor Code</u>	229 (0X00E5)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 230 (0X00E6)

<u>Description</u>	WINQUERYOBJECTWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYOBJECTWINDOW
<u>Minor Code</u>	230 (0X00E6)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 231 (0X00E7)

**Description**  
WINQUERYDESKTOPWINDOW POST-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WinQueryDesktopWindow

**Minor Code**  
231 (0X00E7)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 232 (0X00E8)

**Description**  
WINLOADSTRING POST-INVOCATION

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMWIN.WinLoadString

**Minor Code**  
232 (0X00E8)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 233 (0X00E9)

<u>Description</u>	WINLOADMESSAGE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinLoadMessage
<u>Minor Code</u>	233 (0X00E9)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 234 (0X00EA)

<u>Description</u>	WINQUERYVERSION POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinQueryVersion
<u>Minor Code</u>	234 (0X00EA)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 235 (0X00EB)

<u>Description</u>	WININITIALIZE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinInitialize
<u>Minor Code</u>	235 (0X00EB)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 236 (0X00EC)

**Description**                      WINTERMINATE POST-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WinTerminate

**Minor Code**                      236 (0X00EC)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 300 (0X012C)

**Description**                      WINSETWINDOWPOS PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINSETWINDOWPOS

**Minor Code**                      300 (0X012C)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
%FS = %W, CY = %W, CX = %W, Y = %W, X = %W  
  
HWNDINSERTBEHIND = %A, HWND = %A

# PMWIN Major Code: 0X00C2 Minor Code: 301 (0X012D)

<u>Description</u>	WINQUERYWINDOWPOS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWPOS
<u>Minor Code</u>	301 (0X012D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%SWP = %A, HWND = %A

# PMWIN Major Code: 0X00C2 Minor Code: 302 (0X012E)

<u>Description</u>	WINSETMULTWINDOWPOS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINSETMULTWINDOWPOS
<u>Minor Code</u>	302 (0X012E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%CSWP = %W, PSWP = %A %SWP->FS = %W, PSWP->CY = %W PSWP->CX = %W, PSWP->Y = %W, PSWP->X = %W PSWP->HWNDINSERTBEHIND = %A, PSWP->HWND = %A

# PMWIN Major Code: 0X00C2 Minor Code: 303 (0X012F)

<u>Description</u>	WINSETPARENT PRE-INVOCATION
--------------------	-----------------------------

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETPARENT
<b><u>Minor Code</u></b>	303 (0X012F)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FREDRAW = %W, HWNDNEWPARENT = %A, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 304 (0X0130)

<b><u>Description</u></b>	WINSETOWNER PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETOWNER
<b><u>Minor Code</u></b>	304 (0X0130)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWNDNEWOWNER = %A, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 305 (0X0131)

<b><u>Description</u></b>	WINUPDATEWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINUPDATEWINDOW
<b><u>Minor Code</u></b>	305 (0X0131)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	



%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 306 (0X0132)

### Description

WININVALIDATERECT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WININVALIDATERECT

### Minor Code

306 (0X0132)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%FINCLUDECHILDREN = %W, PWRC = %A, HWND = %A

%WRC->XLEFT = %D, PWRC->YBOTTOM = %D

PWRC->XRIGHT = %D , PWRC->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 307 (0X0133)

### Description

WININVALIDATEREGION PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WININVALIDATEREGION

### Minor Code

307 (0X0133)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%FINCLUDECHILDREN = %W, HRGN = %A, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 308 (0X0134)

**Description**

WINDRAWTEXT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDRAWTEXT

**Minor Code**

308 (0X0134)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RGFCMD = %W, CLRBACK = %D, CLRFORE = %D

PRCL = %A, PCHTEXT = %A, CCHTEXT = %W, HPS = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

PCHTEXT -> %S

-----

PMWIN Major Code: 0X00C2 Minor Code: 309 (0X0135)

**Description**

WINVALIDATERECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINVALIDATERECT

**Minor Code**

309 (0X0135)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FINCLUDECHILDREN = %W, PRCL = %A, HWND = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

PMWIN Major Code: 0X00C2 Minor Code: 310 (0X0136)

<b><u>Description</u></b>	WININVALIDATEREGION PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINVALIDATEREGION
<b><u>Minor Code</u></b>	310 (0X0136)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FINCLUDECHILDREN = %W, HRGN = %A, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 311 (0X0137)

<b><u>Description</u></b>	WINWINDOWFROMDC PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINWINDOWFROMDC
<b><u>Minor Code</u></b>	311 (0X0137)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HDC = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 312 (0X0138)

<b><u>Description</u></b>	WINQUERYWINDOWDC PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYWINDOWDC
<b><u>Minor Code</u></b>	312 (0X0138)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 313 (0X0139)

**Description**

WINGETSCREENPS PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINGETSCREENPS

**Minor Code**

313 (0X0139)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 314 (0X013A)

**Description**

WINQUERYUPDATERECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYUPDATERECT

**Minor Code**

314 (0X013A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RCL = %A, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 315 (0X013B)

<b><u>Description</u></b>	WINQUERYUPDATEREGION PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYUPDATEREGION
<b><u>Minor Code</u></b>	315 (0X013B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HRGN = %A, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 316 (0X013C)

<b><u>Description</u></b>	WINEXCLUDEUPDATEREGION PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINEXCLUDEUPDATEREGION
<b><u>Minor Code</u></b>	316 (0X013C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWND = %A, HPS = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 317 (0X013D)

<b><u>Description</u></b>	WINLOCKWINDOWUPDATE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINLOCKWINDOWUPDATE
<b><u>Minor Code</u></b>	317 (0X013D)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%HWNDLOCKUPDATE = %A, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 318 (0X013E)

**Description**

WINLOCKVISREGIONS PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINLOCKVISREGIONS

**Minor Code**

318 (0X013E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FLOCK = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 319 (0X013F)

**Description**

WINBEGINPAINT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINBEGINPAINT

**Minor Code**

319 (0X013F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RCLPAINT = %A, HPS = %A, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 320 (0X0140)

<b><u>Description</u></b>	WINENDPAINT PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINENDPAINT
<b><u>Minor Code</u></b>	320 (0X0140)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HPS = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 321 (0X0141)

<b><u>Description</u></b>	WINGETPS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINGETPS
<b><u>Minor Code</u></b>	321 (0X0141)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 322 (0X0142)

<b><u>Description</u></b>	WINGETCLIPPS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINGETCLIPPS
<b><u>Minor Code</u></b>	322 (0X0142)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%FS = %W, HWNDCLIP = %A, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 323 (0X0143)

**Description**

WINRELEASEPS PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINRELEASEPS

**Minor Code**

323 (0X0143)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HPS = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 324 (0X0144)

**Description**

WINOPENWINDOWDC PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINOPENWINDOWDC

**Minor Code**

324 (0X0144)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 325 (0X0145)



**Description**

WINSCROLLWINDOW PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSCROLLWINDOW

**Minor Code**

325 (0X0145)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RGFSW = %W, PRCLUPDATE = %A, HRGNUPDATE = %A

PRCLCLIP = %A, PRCLSCROLL = %A, DY = %W, DX = %W

HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 326 (0X0146)

**Description**

WINFILLRECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINFILLRECT

**Minor Code**

326 (0X0146)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LCOLOR = %D, PRCL = %A, HPS = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 327 (0X0147)

**Description**

WINENABLEWINDOWUPDATE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINENABLEWINDOWUPDATE

**Minor Code**

327 (0X0147)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FENABLE = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 400 (0X0190)

**Description**

WINSETWINDOWPOS POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinSetWindowPos

**Minor Code**

400 (0X0190)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 401 (0X0191)

**Description**

WINQUERYWINDOWPOS POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWPOS

**Minor Code**

401 (0X0191)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 402 (0X0192)

<u>Description</u>	WINSETMULTWINDOWPOS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinSetMultWindowPos
<u>Minor Code</u>	402 (0X0192)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 403 (0X0193)

<u>Description</u>	WINUPDATEWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINUPDATEWINDOW
<u>Minor Code</u>	403 (0X0193)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 404 (0X0194)

<u>Description</u>	WININVALIDATERECT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinInvalidateRect

<b><u>Minor Code</u></b>	404 (0X0194)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 405 (0X0195)

<b><u>Description</u></b>	WININVALIDATEREGION POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinInvalidateRegion
<b><u>Minor Code</u></b>	405 (0X0195)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 406 (0X0196)

<b><u>Description</u></b>	WININVERTRECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWININVERTRECT
<b><u>Minor Code</u></b>	406 (0X0196)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 407 (0X0197)

<u>Description</u>	WINDRAWBITMAP POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinDrawBitmap
<u>Minor Code</u>	407 (0X0197)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 408 (0X0198)

<u>Description</u>	WINDRAWTEXT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.DRAWTEXT
<u>Minor Code</u>	408 (0X0198)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 409 (0X0199)

<u>Description</u>	WINDRAWBORDER POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinDrawBorder

**Minor Code** 409 (0X0199)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 410 (0X019A)

**Description** WINVALIDATERECT POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WinValidateRect

**Minor Code** 410 (0X019A)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 411 (0X019B)

**Description** WINVALIDATEREGION POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WinValidateRegion

**Minor Code** 411 (0X019B)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 412 (0X019C)

<u>Description</u>	WINWINDOWFROMDC POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINWINDOWFROMDC
<u>Minor Code</u>	412 (0X019C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 413 (0X019D)

<u>Description</u>	WINQUERYWINDOWDC POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWDC
<u>Minor Code</u>	413 (0X019D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 414 (0X019E)

<u>Description</u>	WINGETSCREENPS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETSCREENPS

**Minor Code** 414 (0X019E)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 415 (0X019F)

**Description** WINQUERYUPDATERECT POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYUPDATERECT

**Minor Code** 415 (0X019F)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 416 (0X01A0)

**Description** WINQUERYUPDATEREGION POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYUPDATEREGION

**Minor Code** 416 (0X01A0)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

DX = %W, AX = %W



-----

## PMWIN Major Code: 0X00C2 Minor Code: 417 (0X01A1)

<u>Description</u>	WINEXCLUDEUPDATEREGION POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINEXCLUDEUPDATEREGION
<u>Minor Code</u>	417 (0X01A1)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 418 (0X01A2)

<u>Description</u>	WINLOCKWINDOWUPDATE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOCKWINDOWUPDATE
<u>Minor Code</u>	418 (0X01A2)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 419 (0X01A3)

<u>Description</u>	WINLOCKVISREGIONS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinLockVisRegions

<b><u>Minor Code</u></b>	419 (0X01A3)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 500 (0X01F4)

<b><u>Description</u></b>	WINBEGINENUMWINDOWS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINBEGINENUMWINDOWS
<b><u>Minor Code</u></b>	500 (0X01F4)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 501 (0X01F5)

<b><u>Description</u></b>	WINGETNEXTWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINGETNEXTWINDOW
<b><u>Minor Code</u></b>	501 (0X01F5)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	%HENUM = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 502 (0X01F6)

<u>Description</u>	WINENDENUMWINDOWS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINENDENUMWINDOWS
<u>Minor Code</u>	502 (0X01F6)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	%HENUM = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 503 (0X01F7)

<u>Description</u>	WINWINDOWFROMPOINT PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINWINDOWFROMPOINT
<u>Minor Code</u>	503 (0X01F7)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	
	%FLOCK = %W, FCHILDREN = %W
	PPTL = %A, HWND = %A
	%TL->X = %D, PPTL->Y = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 504 (0X01F8)

<u>Description</u>	WINMAPWINDOWPOINTS PRE-INVOCATION
--------------------	-----------------------------------

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINMAPWINDOWPOINTS
<b><u>Minor Code</u></b>	504 (0X01F8)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%CWPT = %W, PRGPTL = %A, HWNDTO = %A, HWNDFROM = %A %RGPTL->X = %D, PRGPTL->Y = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 600 (0X0258)

<b><u>Description</u></b>	WINSETACTIVEWINDOW POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETACTIVEWINDOW
<b><u>Minor Code</u></b>	600 (0X0258)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 601 (0X0259)

<b><u>Description</u></b>	WINSUBCLASSWINDOW POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSUBCLASSWINDOW
<b><u>Minor Code</u></b>	601 (0X0259)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 602 (0X025A)

**Description**

WINQUERYCLASSNAME POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCLASSNAME

**Minor Code**

602 (0X025A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 603 (0X025B)

**Description**

WINQUERYCLASSINFO POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinQueryClassInfo

**Minor Code**

603 (0X025B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 604 (0X025C)

<b><u>Description</u></b>	WINQUERYACTIVEWINDOW POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYACTIVEWINDOW
<b><u>Minor Code</u></b>	604 (0X025C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 605 (0X025D)

<b><u>Description</u></b>	WINISTHREADACTIVE POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINISTHREADACTIVE
<b><u>Minor Code</u></b>	605 (0X025D)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 606 (0X025E)

<b><u>Description</u></b>	WINQUERYSYSMODALWINDOW POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYSYSMODALWINDOW
<b><u>Minor Code</u></b>	606 (0X025E)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 607 (0X025F)

**Description**

WINSETSYSMODALWINDOW POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETSYSMODALWINDOW

**Minor Code**

607 (0X025F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 608 (0X0260)

**Description**

WINLOCKWINDOW POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinLockWindow

**Minor Code**

608 (0X0260)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 609 (0X0261)

<u>Description</u>	WINREGISTERWINDOWDESTROY POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINREGISTERWINDOWDESTROY
<u>Minor Code</u>	609 (0X0261)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 610 (0X0262)

<u>Description</u>	WINQUERYWINDOWLOCKCOUNT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWLOCKCOUNT
<u>Minor Code</u>	610 (0X0262)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 611 (0X0263)

<u>Description</u>	WINQUERYWINDOWUSHORT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWVALUE
<u>Minor Code</u>	611 (0X0263)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.



**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 612 (0X0264)

**Description**

WINSETWINDOWUSHORT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWVALUE

**Minor Code**

612 (0X0264)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 613 (0X0265)

**Description**

WINQUERYWINDOWUSHORT/WINQUERYWINDOWULONG/WINQUERYWINDOWPTR POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYWINDOWVALUE

**Minor Code**

613 (0X0265)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 614 (0X0266)

<b><u>Description</u></b>	WINSETWINDOWUSHORT/WINSETWINDOWULONG/WINSETWINDOWPTR POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWVALUE
<b><u>Minor Code</u></b>	614 (0X0266)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 615 (0X0267)

<b><u>Description</u></b>	WINSETWINDOWBITS POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETWINDOWBITS
<b><u>Minor Code</u></b>	615 (0X0267)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 650 (0X028A)

<b><u>Description</u></b>	WINBEGINENUMWINDOWS POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINBEGINENUMWINDOWS
<b><u>Minor Code</u></b>	650 (0X028A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 651 (0X028B)

**Description**

WINGETNEXTWINDOW POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETNEXTWINDOW

**Minor Code**

651 (0X028B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 652 (0X028C)

**Description**

WINENDENUMWINDOWS POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINENDENUMWINDOWS

**Minor Code**

652 (0X028C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 653 (0X028D)

<u>Description</u>	WINWINDOWFROMPOINT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINWINDOWFROMPOINT
<u>Minor Code</u>	653 (0X028D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 654 (0X028E)

<u>Description</u>	WINMAPWINDOWPOINTS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAPWINDOWPOINTS
<u>Minor Code</u>	654 (0X028E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 700 (0X02BC)

<u>Description</u>	WINQUERYQUEUEINFO PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYQUEUEINFO
<u>Minor Code</u>	700 (0X02BC)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.

**Traced Parameters**

%CBCOPY = %W, PMQI = %A, HMQ = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 701 (0X02BD)

**Description**

WINCREATMSGQUEUE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCREATMSGQUEUE

**Minor Code**

701 (0X02BD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CMSG = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 702 (0X02BE)

**Description**

WINDESTROYMSGQUEUE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYMSGQUEUE

**Minor Code**

702 (0X02BE)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HMQ = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 703 (0X02BF)

<b><u>Description</u></b>	WINGETMSG PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINGETMSG
<b><u>Minor Code</u></b>	703 (0X02BF)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%MSGFILTERLAST = %W, MSGFILTERFIRST = %W HWNDFILTER = %A, PQMSG = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 704 (0X02C0)

<b><u>Description</u></b>	WINPEEKMSG PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINPEEKMSG
<b><u>Minor Code</u></b>	704 (0X02C0)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FS = %W, MSGFILTERLAST = %W, MSGFILTERFIRST = %W HWNDFILTER = %A, PQMSG = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 705 (0X02C1)

<b><u>Description</u></b>	WINDISPATCHMSG PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINDISPATCHMSG
<b><u>Minor Code</u></b>	705 (0X02C1)
<b><u>Trace Groups</u></b>	

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%QMSG = %A

%QMSG->HWND = %A, PQMSG->MSG = %W

PQMSG->MP1 = %D, PQMSG->MP2 = %D, PQMSG->TIME = %D

PQMSG->PTL.X = %D, PQMSG->PTL.Y = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 706 (0X02C2)

**Description**

WINPOSTMSG PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINPOSTMSG

**Minor Code**

706 (0X02C2)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 707 (0X02C3)

**Description**

WININSENDMSG PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WININSENDMSG

**Minor Code**

707 (0X02C3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HAB = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 708 (0X02C4)

### Description

WINBROADCASTMSG PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINBROADCASTMSG

### Minor Code

708 (0X02C4)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%RGF = %W, MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 709 (0X02C5)

### Description

WINWAITMSG PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINWAITMSG

### Minor Code

709 (0X02C5)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%MSGLAST = %W, MSGFIRST = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 710 (0X02C6)

### Description

WINQUERYQUEUESTATUS PRE-INVOCATION



<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYQUEUESTATUS
<b><u>Minor Code</u></b>	710 (0X02C6)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWNDDESKTOP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 711 (0X02C7)

<b><u>Description</u></b>	WINPOSTQUEUEMSG PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINPOSTQUEUEMSG
<b><u>Minor Code</u></b>	711 (0X02C7)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%MP2 = %D, MP1 = %D, MSG = %W, HMQ = %A

## PMWIN Major Code: 0X00C2 Minor Code: 712 (0X02C8)

<b><u>Description</u></b>	WINQUERYMSGPOS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYMSGPOS
<b><u>Minor Code</u></b>	712 (0X02C8)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

%TL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 713 (0X02C9)

### Description

WINQUERYMSGTIME PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYMSGTIME

### Minor Code

713 (0X02C9)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%HAB = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 714 (0X02CA)

### Description

WINMSGSEMWAIT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINMSGSEMWAIT

### Minor Code

714 (0X02CA)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%DTTIMEOUT = %D, HSEM = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 715 (0X02CB)

### Description

WINMSGMUXSEMWAIT PRE-INVOCATION

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINMSGMUXSEMWAIT
<b><u>Minor Code</u></b>	715 (0X02CB)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%DTTIMEOUT = %D, PMXSL = %A, PISEMCLEARED = %A %ISEMCLEARED -> %W

## PMWIN Major Code: 0X00C2 Minor Code: 717 (0X02CD)

<b><u>Description</u></b>	WINSETMSGINTEREST PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETMSGINTEREST
<b><u>Minor Code</u></b>	717 (0X02CD)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%CONTROL = %W, MSG_CLASS = %W, HWND = %A

## PMWIN Major Code: 0X00C2 Minor Code: 718 (0X02CE)

<b><u>Description</u></b>	WINSENDMSG PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSENDMSG
<b><u>Minor Code</u></b>	718 (0X02CE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 751 (0X02EF)

**Description**

WINSENDQUEUEMSG PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSENDQUEUEMSG

**Minor Code**

751 (0X02EF)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HMQ = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 752 (0X02F0)

**Description**

CREATEQUEUE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.CREATEQUEUE

**Minor Code**

752 (0X02F0)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CMSGS = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 753 (0X02F1)

<b><u>Description</u></b>	FREEQUEUE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.FREEQUEUE
<b><u>Minor Code</u></b>	753 (0X02F1)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%SMQ = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 754 (0X02F2)

<b><u>Description</u></b>	CLEARQUEUEGLOBALS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.CLEARQUEUEGLOBALS
<b><u>Minor Code</u></b>	754 (0X02F2)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%SMQ = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 755 (0X02F3)

<b><u>Description</u></b>	REASSIGNINPUT PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.REASSIGNINPUT
<b><u>Minor Code</u></b>	755 (0X02F3)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%SMQ = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 756 (0X02F4)

**Description**

ASSOCIATEQUEUE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.ASSOCIATEQUEUE

**Minor Code**

756 (0X02F4)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%SMQ = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 757 (0X02F5)

**Description**

READMESSAGE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.READMESSAGE

**Minor Code**

757 (0X02F5)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FREMOVEMSG = %W, MSGMAXFILTER = %W, MSGMINFILTER = %W

PWNDFILTER = %W, LPMSG = %A, SMQ = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 800 (0X0320)

<b><u>Description</u></b>	WINQUERYQUEUEINFO POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYQUEUEINFO
<b><u>Minor Code</u></b>	800 (0X0320)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 801 (0X0321)

<b><u>Description</u></b>	WINCREATEMSGQUEUE POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinCreateMsgQueue
<b><u>Minor Code</u></b>	801 (0X0321)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 802 (0X0322)

<b><u>Description</u></b>	WINDESTROYMSGQUEUE POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinDestroyMsgQueue
<b><u>Minor Code</u></b>	802 (0X0322)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 803 (0X0323)

**Description**

WINCANCELSHUTDOWN POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinCancelShutdown

**Minor Code**

803 (0X0323)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 804 (0X0324)

**Description**

WINGETMSG/WINPEEKMSG POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETMSG

**Minor Code**

804 (0X0324)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 805 (0X0325)



<u>Description</u>	WINPEEKMSG POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINPEEKMSG
<u>Minor Code</u>	805 (0X0325)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 806 (0X0326)

<u>Description</u>	WINDISPATCHMSG POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINDISPATCHMSG
<u>Minor Code</u>	806 (0X0326)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 807 (0X0327)

<u>Description</u>	WINPOSTMSG/WINPOSTQUEUEMSG POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINPOSTMSG
<u>Minor Code</u>	807 (0X0327)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 808 (0X0328)

**Description**

WININSENDMSG POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWININSENDMSG

**Minor Code**

808 (0X0328)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 809 (0X0329)

**Description**

WINBROADCASTMSG POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINBROADCASTMSG

**Minor Code**

809 (0X0329)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 810 (0X032A)

<b><u>Description</u></b>	WINWAITMSG POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINWAITMSG
<b><u>Minor Code</u></b>	810 (0X032A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 811 (0X032B)

<b><u>Description</u></b>	WINQUERYQUEUESTATUS POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYQUEUESTATUS
<b><u>Minor Code</u></b>	811 (0X032B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 812 (0X032C)

<b><u>Description</u></b>	WINPOSTQUEUEMSG POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINPOSTQUEUEMSG
<b><u>Minor Code</u></b>	812 (0X032C)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 813 (0X032D)

**Description**                      WINQUERYMSGPOS POST-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYMSGPOS

**Minor Code**                      813 (0X032D)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 814 (0X032E)

**Description**                      WINQUERYMSGTIME POST-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYMSGTIME

**Minor Code**                      814 (0X032E)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 815 (0X032F)

<u>Description</u>	WINMSGSEMWAIT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINMSGSEMWAIT
<u>Minor Code</u>	815 (0X032F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 816 (0X0330)

<u>Description</u>	WINMSGSEMWAIT/WINMSGMUXSEMWAIT POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINMSGMUXSEMWAIT
<u>Minor Code</u>	816 (0X0330)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 817 (0X0331)

<u>Description</u>	WINSETMSGINTEREST POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinSetMsgInterest
<u>Minor Code</u>	817 (0X0331)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 818 (0X0332)

**Description**

WINSENDMSG/WINSENDQUEUEMSG POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSENDMSG

**Minor Code**

818 (0X0332)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 900 (0X0384)

**Description**

WINSETFOCUS PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSETFOCUS

**Minor Code**

900 (0X0384)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWNDSETFOCUS = %A, HWNDDESKTOP = %A

---

## PMWIN Major Code: 0X00C2 Minor Code: 901 (0X0385)

<b><u>Description</u></b>	WINFOCUSCHANGE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINFOCUSCHANGE
<b><u>Minor Code</u></b>	901 (0X0385)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FSFOCUSCHANGE = %W, HWNDSETFOCUS = %A HWNDDESKTOP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 902 (0X0386)

<b><u>Description</u></b>	WINSETCAPTURE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETCAPTURE
<b><u>Minor Code</u></b>	902 (0X0386)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWND = %A, HWNDDESKTOP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 903 (0X0387)

<b><u>Description</u></b>	WINQUERYCAPTURE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCAPTURE
<b><u>Minor Code</u></b>	903 (0X0387)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FLOCK = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 904 (0X0388)

**Description**

WINQUERYFOCUS PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYFOCUS

**Minor Code**

904 (0X0388)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FLOCK = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 905 (0X0389)

**Description**

WINSETACTIVEWINDOW PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSETACTIVEWINDOW

**Minor Code**

905 (0X0389)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWND = %A, HWNDDESKTOP = %A

-----



# PMWIN Major Code: 0X00C2 Minor Code: 906 (0X038A)

<u>Description</u>	WINISTHREADACTIVE PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINISTHREADACTIVE
<u>Minor Code</u>	906 (0X038A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%HAB = %A

# PMWIN Major Code: 0X00C2 Minor Code: 907 (0X038B)

<u>Description</u>	WINGETKEYSTATE PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINGETKEYSTATE
<u>Minor Code</u>	907 (0X038B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%VKEY = %W, HWNDDESKTOP = %A

# PMWIN Major Code: 0X00C2 Minor Code: 908 (0X038C)

<u>Description</u>	WINGETPHYSKEYSTATE PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINGETPHYSKEYSTATE
<u>Minor Code</u>	908 (0X038C)

**Trace Groups**                      No groups assigned.

**Trace Types**                        No types assigned.

**Traced Parameters**

  %SC = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 909 (0X038D)

**Description**                            WINENABLEPHYSINPUT PRE-INVOCATION

**Tracepoint**                            Public symbol defined dynamic tracepoint: PMWIN.WINENABLEPHYSINPUT

**Minor Code**                            909 (0X038D)

**Trace Groups**                        No groups assigned.

**Trace Types**                        No types assigned.

**Traced Parameters**

  %FENABLE = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 910 (0X038E)

**Description**                            WINISPHYSINPUTENABLED PRE-INVOCATION

**Tracepoint**                            Public symbol defined dynamic tracepoint: PMWIN.WINISPHYSINPUTENABLED

**Minor Code**                            910 (0X038E)

**Trace Groups**                        No groups assigned.

**Trace Types**                        No types assigned.

**Traced Parameters**

  %HWNDDESKTOP = %A

-----

# PMWIN Major Code: 0X00C2 Minor Code: 911 (0X038F)

Description	WINSETKEYBOARDSTATETABLE PRE-INVOCATION
Tracepoint	Public symbol defined dynamic tracepoint: PMWIN.WINSETKEYBOARDSTATETABLE
Minor Code	911 (0X038F)
Trace Groups	No groups assigned.
Trace Types	No types assigned.
Traced Parameters	%FSET = %W, PKEYSTATETABLE = %A, HWNDDESKTOP = %A %KEYSTATETABLE -> %B

# PMWIN Major Code: 0X00C2 Minor Code: 912 (0X0390)

Description	WINTRACKRECT PRE-INVOCATION
Tracepoint	Public symbol defined dynamic tracepoint: PMWIN.WINTRACKRECT
Minor Code	912 (0X0390)
Trace Groups	No groups assigned.
Trace Types	No types assigned.
Traced Parameters	%TI = %A, HPS = %A, HWND = %A %TI->CXBORDER = %W, PTI->CYBORDER = %W PTI->CXGRID = %W, PTI->CYGRID = %W PTI->CXKEYBOARD = %W, PTI->CYKEYBOARD = %W PTI->RCLTRACK = %A, PTI->RCLBOUNDARY = %A PTI->PTLMINTRACKSIZE = %A, PTI->PTLMAXTRACKSIZE = %A PTI->FS = %W

# PMWIN Major Code: 0X00C2 Minor Code: 913 (0X0391)

<u>Description</u>	WINSHOWTRACKRECT PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINSHOWTRACKRECT
<u>Minor Code</u>	913 (0X0391)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%FSHOW = %W, HWND = %A

# PMWIN Major Code: 0X00C2 Minor Code: 1000 (0X03E8)

<u>Description</u>	WINSETFOCUS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETFOCUS
<u>Minor Code</u>	1000 (0X03E8)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

# PMWIN Major Code: 0X00C2 Minor Code: 1001 (0X03E9)

<u>Description</u>	WINFOCUSCHANGE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINFOCUSCHANGE
<u>Minor Code</u>	

1001 (0X03E9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1002 (0X03EA)

**Description**

WINSETCAPTURE POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETCAPTURE

**Minor Code**

1002 (0X03EA)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1003 (0X03EB)

**Description**

WINQUERYCAPTURE POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCAPTURE

**Minor Code**

1003 (0X03EB)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1004 (0X03EC)

<u>Description</u>	WINQUERYFOCUS/WINQUERYSYSMODALWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYFOCUS
<u>Minor Code</u>	1004 (0X03EC)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1005 (0X03ED)

<u>Description</u>	WINGETKEYSTATE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETKEYSTATE
<u>Minor Code</u>	1005 (0X03ED)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1006 (0X03EE)

<u>Description</u>	WINGETPHYSKEYSTATE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETPHYSKEYSTATE

<b><u>Minor Code</u></b>	1006 (0X03EE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1007 (0X03EF)

<b><u>Description</u></b>	WINENABLEPHYSINPUT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINENABLEPHYSINPUT
<b><u>Minor Code</u></b>	1007 (0X03EF)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1008 (0X03F0)

<b><u>Description</u></b>	WINISPHYSINPUTENABLED POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINISPHYSINPUTENABLED
<b><u>Minor Code</u></b>	1008 (0X03F0)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1009 (0X03F1)

Description	WINSETKEYBOARDSTATETABLE POST-INVOCATION
Tracepoint	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETKEYBOARDSTATETABLE
Minor Code	1009 (0X03F1)
Trace Groups	No groups assigned.
Trace Types	No types assigned.
Traced Parameters	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1100 (0X044C)

Description	WINLOADDLG PRE-INVOCATION
Tracepoint	Public symbol defined dynamic tracepoint: PMWIN.WINLOADDLG
Minor Code	1100 (0X044C)
Trace Groups	No groups assigned.
Trace Types	No types assigned.
Traced Parameters	%CREATEPARAMS = %A, IDDLG = %W, HMOD = %W PFNDLGPROC = %A, HWNDOWNER = %A, HWNDPARENT = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1101 (0X044D)

Description	WINDLGBOX PRE-INVOCATION
Tracepoint	



Public symbol defined dynamic tracepoint: PMWIN.WINDLGBOX

**Minor Code**

1101 (0X044D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CREATEPARAMS = %A, IDDLG = %W, HMOD = %W

PFNDLGPROC = %A, HWNDOWNER = %A, HWNDPARENT = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1102 (0X044E)

**Description**

WINDISMISSDLG PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDISMISSDLG

**Minor Code**

1102 (0X044E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%USRESULT = %W, HWNDIDL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1103 (0X044F)

**Description**

WINSETDLGITEMSHORT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSETDLGITEMSHORT

**Minor Code**

1103 (0X044F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FSIGNED = %W, USVALUE = %W, IDITEM = %W, HWNDDLG = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1104 (0X0450)

### Description

WINQUERYDLGITEMSHORT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYDLGITEMSHORT

### Minor Code

1104 (0X0450)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%FSIGNED = %W, PRESULT = %A, IDITEM = %W, HWNDDLG = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1105 (0X0451)

### Description

WINSETDLGITEMTEXT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINSETDLGITEMTEXT

### Minor Code

1105 (0X0451)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%SZTEXT = %A, IDITEM = %W, HWNDDLG = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1106 (0X0452)

### Description

WINQUERYDLGITEMTEXT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYDLGITEMTEXT

**Minor Code**

1106 (0X0452)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CHBUFFER = %A, CCHBUFFERMAX = %W

IDITEM = %W, HWNDDLK = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1107 (0X0453)

**Description**

WINDEFDLGPROC PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDEFDLGPROC

**Minor Code**

1107 (0X0453)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWNDDLK = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1108 (0X0454)

**Description**

WINALARM PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINALARM

**Minor Code**

1108 (0X0454)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RGFTYPE = %W, HWNDDESKTOP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1109 (0X0455)

**Description**

WINMESSAGEBOX PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINMESSAGEBOX

**Minor Code**

1109 (0X0455)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FLSTYLE = %W, IDWINDOW = %W, PSZCAPTION = %A  
PSZTEXT = %A, HWNDOWNER = %A, HWNDPARENT = %A  
PSZCAPTION -> %S  
PSZTEXT -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1110 (0X0456)

**Description**

WINPROCESSDLG PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINPROCESSDLG

**Minor Code**

1110 (0X0456)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWNDIDL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1111 (0X0457)

<u>Description</u>	WINSENDDLGITEMMSG PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINSENDDLGITEMMSG
<u>Minor Code</u>	1111 (0X0457)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %MP2 = %D, MP1 = %D, MSG = %W IDITEM = %W, HWNDDLGL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1112 (0X0458)

<u>Description</u>	WINMAPDLGPOINTS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINMAPDLGPOINTS
<u>Minor Code</u>	1112 (0X0458)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %FCALCWINDOWCOORDS = %W, CWPT = %W PRGWPTL = %A, HWNDDLGL = %A %RGWPTL->X = %D, PRGWPTL->Y = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1113 (0X0459)

<b><u>Description</u></b>	WINSUBSTITUTESTRINGS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSUBSTITUTESTRINGS
<b><u>Minor Code</u></b>	1113 (0X0459)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%SZDST = %A, CCHDSTMAX = %W, PSZSRC = %A, HWND = %A PSZSRC -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1114 (0X045A)

<b><u>Description</u></b>	WINENUMDLGITEM PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINENUMDLGITEM
<b><u>Minor Code</u></b>	1114 (0X045A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FLOCK = %W, CODE = %W, HWND = %A, HWNDDLG = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1115 (0X045B)

<b><u>Description</u></b>	WINCREATEDLG PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINCREATEDLG
<b><u>Minor Code</u></b>	1115 (0X045B)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CREATEPARAMS = %A, PDLGT = %A, PFNDLGPROC = %A

HWNDOWNER = %A, HWNDPARENT = %A

%DLGT->CBTEMPLATE = %W, PDLGT->TYPE = %W

PDLGT->CODEPAGE = %W, PDLGT->OFFADLGTI = %W

PDLGT->FSTEMPLATESTATUS = %W, PDLGT->IITEMFOCUS = %W

PDLGT->COFFPRESPARAMS = %W, PDLGT->ADLGTI[] = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1150 (0X047E)

**Description**

WINEDITWNDPROC PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINEDITWNDPROC

**Minor Code**

1150 (0X047E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1200 (0X04B0)

**Description**

WINLOADDLG POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinLoadDlg

**Minor Code**

1200 (0X04B0)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1201 (0X04B1)

### Description

WINDLGBOX POST-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WinDlgBox

### Minor Code

1201 (0X04B1)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1202 (0X04B2)

### Description

WINDISMISSDLG POST-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WinDismissDlg

### Minor Code

1202 (0X04B2)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1203 (0X04B3)

### Description



WINSETDLGITEMSHORT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETDLGITEMSHORT

**Minor Code**

1203 (0X04B3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1204 (0X04B4)

**Description**

WINQUERYDLGITEMSHORT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYDLGITEMSHORT

**Minor Code**

1204 (0X04B4)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1205 (0X04B5)

**Description**

WINSETDLGITEMTEXT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinSetDlgItemText

**Minor Code**

1205 (0X04B5)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1206 (0X04B6)

**Description**

WINQUERYDLGITEMTEXT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinQueryDlgItemText

**Minor Code**

1206 (0X04B6)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1207 (0X04B7)

**Description**

WINDEFDLGPROC POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinDefDlgProc

**Minor Code**

1207 (0X04B7)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1208 (0X04B8)

<b><u>Description</u></b>	WINALARM POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINALARM
<b><u>Minor Code</u></b>	1208 (0X04B8)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1209 (0X04B9)

<b><u>Description</u></b>	WINMESSAGEBOX POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinMessageBox
<b><u>Minor Code</u></b>	1209 (0X04B9)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1210 (0X04BA)

<b><u>Description</u></b>	WINPROCESSDLG POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinProcessDlg
<b><u>Minor Code</u></b>	1210 (0X04BA)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1211 (0X04BB)

**Description**

WINSENDDLGITMSG POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSENDDLGITMSG

**Minor Code**

1211 (0X04BB)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1212 (0X04BC)

**Description**

WINMAPDLGPOINTS POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAPDLGPOINTS

**Minor Code**

1212 (0X04BC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1213 (0X04BD)

<u>Description</u>	WINSUBSTITUTESTRINGS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSUBSTITUTESTRINGS
<u>Minor Code</u>	1213 (0X04BD)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1214 (0X04BE)

<u>Description</u>	WINENUMDLGITEM POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINENUMDLGITEM
<u>Minor Code</u>	1214 (0X04BE)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1215 (0X04BF)

<u>Description</u>	WINCREATEDLG POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinCreateDlg
<u>Minor Code</u>	1215 (0X04BF)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1300 (0X0514)

**Description**

WINLOADMENU PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINLOADMENU

**Minor Code**

1300 (0X0514)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%IDMENU = %W, HMOD = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1301 (0X0515)

**Description**

WINCREATEMENU PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCREATEMENU

**Minor Code**

1301 (0X0515)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LPMT = %A, HWNDPARENT = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1302 (0X0516)

<b><u>Description</u></b>	WINFLASHWINDOW PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINFLASHWINDOW
<b><u>Minor Code</u></b>	1302 (0X0516)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FFLASH = %W, HWNDFRAME = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1303 (0X0517)

<b><u>Description</u></b>	WINCREATEFRAMECONTROLS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINCREATEFRAMECONTROLS
<b><u>Minor Code</u></b>	1303 (0X0517)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%SZTITLE = %A, PFCDATA = %A HWNDFRAME = %A %FCDATA->CB = %W, PFCDATA->FLCREATEFLAGS = %D PFCDATA->HMODRESOURCES = %W, PFCDATA->IDRESOURCES = %W PSZTITLE -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1304 (0X0518)

<b><u>Description</u></b>	WINFORMATFRAME PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINFORMATFRAME

**Minor Code**

1304 (0X0518)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RCLCLIENT = %A, CSWPMAX = %W, PSWP = %A

PRCLFRAME = %A, HWNDFRAME = %A

%RCLFRAME-&gt;XLEFT = %D, PRCLFRAME-&gt;YBOTTOM = %D

PRCLFRAME-&gt;XRIGHT = %D, PRCLFRAME-&gt;YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1305 (0X0519)

**Description**

WINCALCFRAMERECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCALCFRAMERECT

**Minor Code**

1305 (0X0519)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FCLIENT = %W, PRCL = %A, HWNDFRAME = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1306 (0X051A)

**Description**

WINDRAWBITMAP PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDRAWBITMAP

**Minor Code**

1306 (0X051A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.



**Traced Parameters**

%FS = %W, CLRBACK = %D, CLRFORE = %D  
PPTLDST = %A, PWRCSRC = %A, HBM = %A, HPSDST = %A  
%TLDST->X = %D, PPTLDST->Y = %D  
%WRCSRC->XLEFT = %D, PWRCSRC->YBOTTOM = %D  
PWRCSRC->XRIGHT = %D, PWRCSRC->YTOP = %D

-----

PMWIN Major Code: 0X00C2 Minor Code: 1307 (0X051B)

**Description**

WINDRAWBORDER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDRAWBORDER

**Minor Code**

1307 (0X051B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RGFCMD = %W, CLRBACK = %D, CLRFORE = %D  
CY = %W, CX = %W, PRCL = %A, HPS = %A  
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D  
PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

PMWIN Major Code: 0X00C2 Minor Code: 1308 (0X051C)

**Description**

WINGETMINPOSITION PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINGETMINPOSITION

**Minor Code**

1308 (0X051C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%TL = %A, PSWP = %A, HWND = %A  
%TL->X = %D, PPTL->Y = %D

-----

PMWIN Major Code: 0X00C2 Minor Code: 1309 (0X051D)

**Description**

WINGETMAXPOSITION PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINGETMAXPOSITION

**Minor Code**

1309 (0X051D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%SWP = %A, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1350 (0X0546)

**Description**

WINFRAMEWNDPROC PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINFRAMEWNDPROC

**Minor Code**

1350 (0X0546)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1351 (0X0547)

<u><b>Description</b></u>	WINSCROLLBARWNDPROC PRE-INVOCATION
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMWIN.WINSCROLLBARWNDPROC
<u><b>Minor Code</b></u>	1351 (0X0547)
<u><b>Trace Groups</b></u>	No groups assigned.
<u><b>Trace Types</b></u>	No types assigned.
<u><b>Traced Parameters</b></u>	%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

## PMWIN Major Code: 0X00C2 Minor Code: 1352 (0X0548)

<u><b>Description</b></u>	WINTITLEBARWNDPROC PRE-INVOCATION
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMWIN.WINTITLEBARWNDPROC
<u><b>Minor Code</b></u>	1352 (0X0548)
<u><b>Trace Groups</b></u>	No groups assigned.
<u><b>Trace Types</b></u>	No types assigned.
<u><b>Traced Parameters</b></u>	%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

## PMWIN Major Code: 0X00C2 Minor Code: 1353 (0X0549)

<u><b>Description</b></u>	WINSTATICWNDPROC PRE-INVOCATION
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMWIN.WINSTATICWNDPROC
<u><b>Minor Code</b></u>	1353 (0X0549)
<u><b>Trace Groups</b></u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1354 (0X054A)

**Description**

WINICONTXTWNDPROC PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINICONTXTWNDPROC

**Minor Code**

1354 (0X054A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MP2 = %D, MP1 = %D, MSG = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1355 (0X054B)

**Description**

WINCALLHELPHOOK PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCALLHELPHOOK

**Minor Code**

1355 (0X054B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RCPOSITION = %A, IDSUBTOPIC = %W

IDTOPIC = %W, SMODE = %W

%RCPOSITION->XLEFT = %D, PRCPOSITION->YBOTTOM = %D

PRCPOSITION->XRIGHT = %D, PRCPOSITION->YTOP = %D

-----

PMWIN Major Code: 0X00C2 Minor Code: 1356 (0X054C)

<u>Description</u>	DESTROYLIST PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.DESTROYLIST
<u>Minor Code</u>	1356 (0X054C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%LST = %W, HHEAP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1357 (0X054D)

<u>Description</u>	INSERTLISTITEM PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.INSERTLISTITEM
<u>Minor Code</u>	1357 (0X054D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%CCHINSERT = %W, LPCHINSERT = %A, ILI = %W PLST = %W, HHEAP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1358 (0X054E)

<u>Description</u>	DELETELISTITEM PRE-INVOCATION
--------------------	-------------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.DELETETESTITEM
<u>Minor Code</u>	1358 (0X054E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%ILI = %W, PLST = %W, HHEAP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 1359 (0X054F)

<u>Description</u>	GETLISTITEMLENGTH PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.GETLISTITEMLENGTH
<u>Minor Code</u>	1359 (0X054F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%ILI = %W, PLST = %W, HHEAP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 1360 (0X0550)

<u>Description</u>	GETLISTITEM PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.GETLISTITEM
<u>Minor Code</u>	1360 (0X0550)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	

%LPCHDEST = %A, CCHGET = %W, ICHFIRST = %W

ILI = %W, PLST = %W, HHEAP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1362 (0X0552)

### Description

SETLISTITEM PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.SETLISTITEM

### Minor Code

1362 (0X0552)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%FRESIZE = %W, LPCGSRC = %A, CBSET = %W, IBLI = %W

ILI = %W, PLST = %W, HHEAP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1400 (0X0578)

### Description

WINLOADMENU POST-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOADMENU

### Minor Code

1400 (0X0578)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1401 (0X0579)

<u>Description</u>	WINCREATEMENU POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinCreateMenu
<u>Minor Code</u>	1401 (0X0579)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1402 (0X057A)

<u>Description</u>	WINFLASHWINDOW POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinFlashWindow
<u>Minor Code</u>	1402 (0X057A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1403 (0X057B)

<u>Description</u>	WINCREATEFRAMECONTROLS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATEFRAMECONTROLS
<u>Minor Code</u>	1403 (0X057B)
<u>Trace Groups</u>	No groups assigned.



**Trace Types** No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1404 (0X057C)

**Description** WINFORMATFRAME POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WinFormatFrame

**Minor Code** 1404 (0X057C)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1405 (0X057D)

**Description** WINCALCFRAMERECT POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WinCalcFrameRect

**Minor Code** 1405 (0X057D)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1406 (0X057E)

<b><u>Description</u></b>	WINGETMINPOSITION POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETMINPOSITION
<b><u>Minor Code</u></b>	1406 (0X057E)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1407 (0X057F)

<b><u>Description</u></b>	WINGETMAXPOSITION POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETMAXPOSITION
<b><u>Minor Code</u></b>	1407 (0X057F)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1500 (0X05DC)

<b><u>Description</u></b>	WINSETRECT PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETRECT
<b><u>Minor Code</u></b>	1500 (0X05DC)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%YTOP = %W, XRIGHT = %W, YBOTTOM = %W

XLEFT = %W, PRCL = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1501 (0X05DD)

**Description**

WINISRECTEMPTY PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINISRECTEMPTY

**Minor Code**

1501 (0X05DD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RCL = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1502 (0X05DE)

**Description**

WINCOPYRECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCOPYRECT

**Minor Code**

1502 (0X05DE)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

#### Traced Parameters

%RCLSRC = %A, PRCLDST = %A  
%RCLSRC->XLEFT = %D, PRCLSRC->YBOTTOM = %D  
PRCLSRC->XRIGHT = %D, PRCLSRC->YTOP = %D  
%RCLDST->XLEFT = %D, PRCLDST->YBOTTOM = %D  
PRCLDST->XRIGHT = %D, PRCLDST->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1503 (0X05DF)

#### Description

WINEQUALRECT PRE-INVOCATION

#### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINEQUALRECT

#### Minor Code

1503 (0X05DF)

#### Trace Groups

No groups assigned.

#### Trace Types

No types assigned.

#### Traced Parameters

%RCL2 = %A, PRCL1 = %A  
%RCL2->XLEFT = %D, PRCL2->YBOTTOM = %D  
PRCL2->XRIGHT = %D, PRCL2->YTOP = %D  
%RCL1->XLEFT = %D, PRCL1->YBOTTOM = %D  
PRCL1->XRIGHT = %D, PRCL1->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1504 (0X05E0)

#### Description

WINSETRECTEMPTY PRE-INVOCATION

#### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINSETRECTEMPTY

#### Minor Code

1504 (0X05E0)

#### Trace Groups

No groups assigned.

#### Trace Types

No types assigned.

**Traced Parameters**

%RCL = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1505 (0X05E1)

**Description**

WINOFFSETRECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINOFFSETRECT

**Minor Code**

1505 (0X05E1)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CY = %W, CX = %W, PRCL = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1506 (0X05E2)

**Description**

WININFLATERECT PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WININFLATERECT

**Minor Code**

1506 (0X05E2)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CY = %W, CX = %W, PRCL = %A

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1507 (0X05E3)

### Description

WINPTINRECT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINPTINRECT

### Minor Code

1507 (0X05E3)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%TL = %A, PRCL = %A

%TL->X = %D, PPTL->Y = %D

%RCL->XLEFT = %D, PRCL->YBOTTOM = %D

PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1508 (0X05E4)

### Description

WININTERSECTRECT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WININTERSECTRECT

### Minor Code

1508 (0X05E4)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%RCLSRC2 = %A, PRCLSRC1 = %A, PRCLDST = %A

%RCLSRC2->XLEFT = %D, PRCLSRC2->YBOTTOM = %D

PRCLSRC2->XRIGHT = %D, PRCLSRC2->YTOP = %D

%RCLSRC1->XLEFT = %D, PRCLSRC1->YBOTTOM = %D

PRCLSRC1->XRIGHT = %D, PRCLSRC1->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1509 (0X05E5)

### Description

WINUNIONRECT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINUNIONRECT

### Minor Code

1509 (0X05E5)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%RCLSRC2 = %A, PRCLSRC1 = %A, PRCLDST = %A

%RCLSRC2->XLEFT = %D, PRCLSRC2->YBOTTOM = %D

PRCLSRC2->XRIGHT = %D, PRCLSRC2->YTOP = %D

%RCLSRC1->XLEFT = %D, PRCLSRC1->YBOTTOM = %D

PRCLSRC1->XRIGHT = %D, PRCLSRC1->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1510 (0X05E6)

### Description

WINSUBTRACTRECT PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINSUBTRACTRECT

### Minor Code

1510 (0X05E6)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%RCLSRC2 = %A, PRCLSRC1 = %A, PRCLDST = %A

%RCLSRC2->XLEFT = %D, PRCLSRC2->YBOTTOM = %D

PRCLSRC2->XRIGHT = %D, PRCLSRC2->YTOP = %D  
%RCLSRC1->XLEFT = %D, PRCLSRC1->YBOTTOM = %D  
PRCLSRC1->XRIGHT = %D, PRCLSRC1->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1511 (0X05E7)

**Description** WININVERTRECT PRE-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WININVERTRECT

**Minor Code** 1511 (0X05E7)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

%RCL = %A, HPS = %A  
%RCL->XLEFT = %D, PRCL->YBOTTOM = %D  
PRCL->XRIGHT = %D, PRCL->YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1512 (0X05E8)

**Description** WINMAKERECT PRE-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WINMAKERECT

**Minor Code** 1512 (0X05E8)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**

%WRC = %A  
%WRC->XLEFT = %W, PWRC->DUMMY1 = %W  
PWRC->YBOTTOM = %W, PWRC->DUMMY2 = %W



PWRC->XRIGHT= %W, PWRC->DUMMY3 = %W

PWRC->YTOP = %W, PWRC->DUMMY4 = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1513 (0X05E9)

### Description

WINMAKEPOINTS PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINMAKEPOINTS

### Minor Code

1513 (0X05E9)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%CWPT = %W, PWPT = %A

%WPT->X = %W, PWPT->DUMMY1 = %W

PWPT->Y = %W, PWPT->DUMMY2 = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1600 (0X0640)

### Description

WINSETRECT POST-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETRECT

### Minor Code

1600 (0X0640)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1601 (0X0641)

<b><u>Description</u></b>	WINISRECTEMPTY POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINISRECTEMPTY
<b><u>Minor Code</u></b>	1601 (0X0641)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1602 (0X0642)

<b><u>Description</u></b>	WINCOPYRECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINCOPYRECT
<b><u>Minor Code</u></b>	1602 (0X0642)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1603 (0X0643)

<b><u>Description</u></b>	WINEQUALRECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINEQUALRECT
<b><u>Minor Code</u></b>	1603 (0X0643)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1604 (0X0644)

**Description**

WINSETRECTEMPTY POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETRECTEMPTY

**Minor Code**

1604 (0X0644)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1605 (0X0645)

**Description**

WINOFFSETRECT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINOFFSETRECT

**Minor Code**

1605 (0X0645)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1606 (0X0646)

<b><u>Description</u></b>	WININFLATERECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWININFLATERECT
<b><u>Minor Code</u></b>	1606 (0X0646)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1607 (0X0647)

<b><u>Description</u></b>	WINPTINRECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINPTINRECT
<b><u>Minor Code</u></b>	1607 (0X0647)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1608 (0X0648)

<b><u>Description</u></b>	WININTERSECTRECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWININTERSECTRECT
<b><u>Minor Code</u></b>	1608 (0X0648)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1609 (0X0649)

**Description**

WINUNIONRECT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINUNIONRECT

**Minor Code**

1609 (0X0649)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1610 (0X064A)

**Description**

WINSUBTRACTRECT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSUBTRACTRECT

**Minor Code**

1610 (0X064A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1611 (0X064B)

<b><u>Description</u></b>	WINMAKERECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAKERECT
<b><u>Minor Code</u></b>	1611 (0X064B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1612 (0X064C)

<b><u>Description</u></b>	WINMAKEPOINTS POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINMAKEPOINTS
<b><u>Minor Code</u></b>	1612 (0X064C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1613 (0X064D)

<b><u>Description</u></b>	WINTRACKRECT POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinTrackRect
<b><u>Minor Code</u></b>	1613 (0X064D)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1614 (0X064E)

**Description**

WINSHOWTRACKRECT POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSHOWTRACKRECT

**Minor Code**

1614 (0X064E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1700 (0X06A4)

**Description**

WINQUERYVERSION PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYVERSION

**Minor Code**

1700 (0X06A4)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HAB = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1701 (0X06A5)

<b><u>Description</u></b>	WININITIALIZE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WININITIALIZE
<b><u>Minor Code</u></b>	1701 (0X06A5)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1702 (0X06A6)

<b><u>Description</u></b>	WINTERMINATE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINTERMINATE
<b><u>Minor Code</u></b>	1702 (0X06A6)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HAB = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1703 (0X06A7)

<b><u>Description</u></b>	WINQUERYSYSTEMATOMTABLE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSTEMATOMTABLE
<b><u>Minor Code</u></b>	1703 (0X06A7)
<b><u>Trace Groups</u></b>	No groups assigned.



**Trace Types**

No types assigned.

**Traced Parameters**

NO PARAMETERS

-----

PMWIN Major Code: 0X00C2 Minor Code: 1704 (0X06A8)

**Description**

WINQUERYSYSVALUE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSVALUE

**Minor Code**

1704 (0X06A8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ISYSVALUE = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1705 (0X06A9)

**Description**

WINSETSYSVALUE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSETSYSVALUE

**Minor Code**

1705 (0X06A9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LVALUE = %D, ISYSVALUE = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1706 (0X06AA)

<b><u>Description</u></b>	WINQUERYSYSCOLOR PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSCOLOR
<b><u>Minor Code</u></b>	1706 (0X06AA)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%LRESERVED = %D, ICOLOR = %D, HWNDDESKTOP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1707 (0X06AB)

<b><u>Description</u></b>	WINSETSYSCOLORS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETSYSCOLORS
<b><u>Minor Code</u></b>	1707 (0X06AB)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%CLR = %A, CCLR = %D, CLRFIRST = %D FLFORMAT = %D, FLOPTIONS = %D, HWNDDESKTOP = %A %CLR -> %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1708 (0X06AC)

<b><u>Description</u></b>	WINCATCH PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINCATCH
<b><u>Minor Code</u></b>	

1708 (0X06AC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CATCHBUF = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1709 (0X06AD)

**Description**

WINTHROW PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINTHROW

**Minor Code**

1709 (0X06AD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%NTHROWBACK = %W, PCATCHBUF = %A

%CATCHBUF->CTCHBF = %W %W %W %W %W %W %W %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1710 (0X06AE)

**Description**

WINGETLASTERROR PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINGETLASTERROR

**Minor Code**

1710 (0X06AE)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HAB = %A

---

## PMWIN Major Code: 0X00C2 Minor Code: 1711 (0X06AF)

### Description

WINGETERRORINFO PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINGETERRORINFO

### Minor Code

1711 (0X06AF)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%HAB = %A

---

## PMWIN Major Code: 0X00C2 Minor Code: 1712 (0X06B0)

### Description

WINFREEERRORINFO PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINFREEERRORINFO

### Minor Code

1712 (0X06B0)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%ERRINFO = %A

%ERRINFO->CBFIXEDERRINFO = %W

PERRINFO->IDERROR = %D, PERRINFO->CDETAILLEVEL = %W

PERRINFO->OFFAOFFSZMSG = %W, PERRINFO->OFFBINARYDATA = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 1713 (0X06B1)

<b><u>Description</u></b>	WINSTARTTIMER PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSTARTTIMER
<b><u>Minor Code</u></b>	1713 (0X06B1)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%DTIMEOUT = %W, IDTIMER = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1714 (0X06B2)

<b><u>Description</u></b>	WINSTOPTIMER PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSTOPTIMER
<b><u>Minor Code</u></b>	1714 (0X06B2)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%IDTIMER = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1715 (0X06B3)

<b><u>Description</u></b>	WINGETCURRENTTIME PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINGETCURRENTTIME
<b><u>Minor Code</u></b>	1715 (0X06B3)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %HAB = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 1750 (0X06D6)

**Description**                      WINSETERRORINFO PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.\_WINSETERRORINFO

**Minor Code**                      1750 (0X06D6)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %RETADDR = %A, ERRCODE = %D, FOPTIONS = %W, ARGS = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 1751 (0X06D7)

**Description**                      WINSETLASTERROR PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINSETLASTERROR

**Minor Code**                      1751 (0X06D7)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %RETADDR = %A, ERRCODE = %D

-----

PMWIN Major Code: 0X00C2 Minor Code: 1800 (0X0708)

<u>Description</u>	WINQUERYSYSVALUE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYSYSVALUE
<u>Minor Code</u>	1800 (0X0708)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1801 (0X0709)

<u>Description</u>	WINSETSYSVALUE POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETSYSVALUE
<u>Minor Code</u>	1801 (0X0709)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1802 (0X070A)

<u>Description</u>	WINQUERYSYSCOLOR POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinQuerySysColor
<u>Minor Code</u>	1802 (0X070A)
<u>Trace Groups</u>	No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1803 (0X070B)

**Description** WINSETSYSCOLORS POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.WinSetSysColors

**Minor Code** 1803 (0X070B)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1804 (0X070C)

**Description** WINCATCH POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.POSTWINCATCH

**Minor Code** 1804 (0X070C)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1805 (0X070D)



<b><u>Description</u></b>	WINTHROW POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINTHROW
<b><u>Minor Code</u></b>	1805 (0X070D)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1806 (0X070E)

<b><u>Description</u></b>	WINGETLASTERROR POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinGetLastError
<b><u>Minor Code</u></b>	1806 (0X070E)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1807 (0X070F)

<b><u>Description</u></b>	WINGETERRORINFO POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinGetErrorInfo
<b><u>Minor Code</u></b>	1807 (0X070F)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1808 (0X0710)

**Description**

WINFREEERRORINFO POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinFreeErrorInfo

**Minor Code**

1808 (0X0710)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1809 (0X0711)

**Description**

WINSTARTTIMER POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSTARTTIMER

**Minor Code**

1809 (0X0711)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1810 (0X0712)

<b><u>Description</u></b>	WINSTOPTIMER POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSTOPTIMER
<b><u>Minor Code</u></b>	1810 (0X0712)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1811 (0X0713)

<b><u>Description</u></b>	WINGETCURRENTTIME POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETCURRENTTIME
<b><u>Minor Code</u></b>	1811 (0X0713)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1900 (0X076C)

<b><u>Description</u></b>	WINLOADACCELTABLE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINLOADACCELTABLE
<b><u>Minor Code</u></b>	1900 (0X076C)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%IDACCELTABLE = %W, HMOD = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1901 (0X076D)

**Description**

WINCREATEACCELTABLE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCREATEACCELTABLE

**Minor Code**

1901 (0X076D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ACCELTABLE = %A

%ACCELTABLE->CACCEL = %W, PACCELTABLE->CODEPAGE = %W

PACCELTABLE->FS = %W, PACCELTABLE->KEY = %W

PACCELTABLE->CMD= %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1902 (0X076E)

**Description**

WINDESTROYACCELTABLE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYACCELTABLE

**Minor Code**

1902 (0X076E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HACCEL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1903 (0X076F)

<u>Description</u>	WINCOPYACCELTABLE PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINCOPYACCELTABLE
<u>Minor Code</u>	1903 (0X076F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%CBCOPYMAX = %W, PACCELTABLE = %A, HACCEL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1904 (0X0770)

<u>Description</u>	WINTRANSLATEACCEL PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINTRANSLATEACCEL
<u>Minor Code</u>	1904 (0X0770)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%QMSG = %A, HACCEL = %A, HWND = %A %QMSG->HWND = %A, PQMSG->MSG = %W, PQMSG->MP1 = %D PQMSG->MP2 = %D, PQMSG->TIME = %D, PQMSG->PTL.X = %D PQMSG->PTL.Y = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1905 (0X0771)

<b><u>Description</u></b>	WINSETACCELTABLE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETACCELTABLE
<b><u>Minor Code</u></b>	1905 (0X0771)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWNDFRAME = %A, HACCEL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 1906 (0X0772)

<b><u>Description</u></b>	WINQUERYACCELTABLE PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYACCELTABLE
<b><u>Minor Code</u></b>	1906 (0X0772)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HWNDFRAME = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2000 (0X07D0)

<b><u>Description</u></b>	WINLOADACCELTABLE POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOADACCELTABLE
<b><u>Minor Code</u></b>	2000 (0X07D0)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2001 (0X07D1)

**Description** WINCREATEACCELTABLE POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATEACCELTABLE

**Minor Code** 2001 (0X07D1)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2002 (0X07D2)

**Description** WINDESTROYACCELTABLE POST-INVOCATION

**Tracepoint** Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYACCELTABLE

**Minor Code** 2002 (0X07D2)

**Trace Groups** No groups assigned.

**Trace Types** No types assigned.

**Traced Parameters**  
  
DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2003 (0X07D3)

<b><u>Description</u></b>	WINCOPYACCELTABLE POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINCOPYACCELTABLE
<b><u>Minor Code</u></b>	2003 (0X07D3)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2004 (0X07D4)

<b><u>Description</u></b>	WINTRANSLATEACCEL POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINTRANSLATEACCEL
<b><u>Minor Code</u></b>	2004 (0X07D4)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2005 (0X07D5)

<b><u>Description</u></b>	WINSETACCELTABLE POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETACCELTABLE
<b><u>Minor Code</u></b>	2005 (0X07D5)
<b><u>Trace Groups</u></b>	No groups assigned.



**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 2006 (0X07D6)

**Description**

WINQUERYACCELTABLE POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYACCELTABLE

**Minor Code**

2006 (0X07D6)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 2100 (0X0834)

**Description**

WINOPENCLIPBRD PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINOPENCLIPBRD

**Minor Code**

2100 (0X0834)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HAB = %A

---

## PMWIN Major Code: 0X00C2 Minor Code: 2101 (0X0835)

<b><u>Description</u></b>	WINCLOSECLIPBRD PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINCLOSECLIPBRD
<b><u>Minor Code</u></b>	2101 (0X0835)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HAB = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2102 (0X0836)

<b><u>Description</u></b>	WINEMPTYCLIPBRD PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINEMPTYCLIPBRD
<b><u>Minor Code</u></b>	2102 (0X0836)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HAB = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2103 (0X0837)

<b><u>Description</u></b>	WINSETCLIPBRDOWNER PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETCLIPBRDOWNER
<b><u>Minor Code</u></b>	2103 (0X0837)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2104 (0X0838)

**Description**

WINQUERYCLIPBRDOWNER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDOWNER

**Minor Code**

2104 (0X0838)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FLOCK = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2105 (0X0839)

**Description**

WINSETCLIPBRDDATA PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSETCLIPBRDDATA

**Minor Code**

2105 (0X0839)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RGFFMTINFO = %W, FMT = %W, ULDATA = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2106 (0X083A)

<u>Description</u>	WINQUERYCLIPBRDDATA PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDDATA
<u>Minor Code</u>	2106 (0X083A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%FMT = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2107 (0X083B)

<u>Description</u>	WINENUMCLIPBRDFMTS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINENUMCLIPBRDFMTS
<u>Minor Code</u>	2107 (0X083B)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%FMT = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2108 (0X083C)

<u>Description</u>	WINQUERYCLIPBRDFMTINFO PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDFMTINFO
<u>Minor Code</u>	2108 (0X083C)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %RGFFMTINFO = %A, FMT = %W

   %RGFFMTINFO -> %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 2109 (0X083D)

**Description**                      WINSETCLIPBRDVIEWER PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINSETCLIPBRDVIEWER

**Minor Code**                      2109 (0X083D)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %HWNDNEWCLIPVIEWER = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2110 (0X083E)

**Description**                      WINQUERYCLIPBRDVIEWER PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCLIPBRDVIEWER

**Minor Code**                      2110 (0X083E)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %FLOCK = %W

-----

# PMWIN Major Code: 0X00C2 Minor Code: 2200 (0X0898)

<u>Description</u>	WINOPENCLIPBRD POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinOpenClipbrd
<u>Minor Code</u>	2200 (0X0898)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

# PMWIN Major Code: 0X00C2 Minor Code: 2201 (0X0899)

<u>Description</u>	WINCLOSECLIPBRD POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinCloseClipbrd
<u>Minor Code</u>	2201 (0X0899)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

# PMWIN Major Code: 0X00C2 Minor Code: 2202 (0X089A)

<u>Description</u>	WINEMPTYCLIPBRD POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinEmptyClipbrd
<u>Minor Code</u>	2202 (0X089A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2203 (0X089B)

**Description**

WINSETCLIPBRDOWNER POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinSetClipbrdOwner

**Minor Code**

2203 (0X089B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2204 (0X089C)

**Description**

WINQUERYCLIPBRDOWNER/WINQUERYCIPBRDVIEWER/WINQUERYFOCUS/WINQUERYSYSMODALWINDOW POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCLIPBRDOWNER

**Minor Code**

2204 (0X089C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2205 (0X089D)

<u>Description</u>	WINSETCLIPBRDDATA POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinSetClipbrdData
<u>Minor Code</u>	2205 (0X089D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2206 (0X089E)

<u>Description</u>	WINQUERYCLIPBRDDATA POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinQueryClipbrdData
<u>Minor Code</u>	2206 (0X089E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2207 (0X089F)

<u>Description</u>	WINENUMCLIPBRDFMTS POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinEnumClipbrdFmts



<b><u>Minor Code</u></b>	2207 (0X089F)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2208 (0X08A0)

<b><u>Description</u></b>	WINQUERYCLIPBRDFMTINFO POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinQueryClipbrdFmtInfo
<b><u>Minor Code</u></b>	2208 (0X08A0)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2209 (0X08A1)

<b><u>Description</u></b>	WINSETCLIPBRDVIEWER POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinSetClipbrdViewer
<b><u>Minor Code</u></b>	2209 (0X08A1)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2210 (0X08A2)

<u>Description</u>	WINQUERYCLIPBRDVIEWER POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCLIPBRDVIEWER
<u>Minor Code</u>	2210 (0X08A2)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2300 (0X08FC)

<u>Description</u>	WINDESTROYCURSOR PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYCURSOR
<u>Minor Code</u>	2300 (0X08FC)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2301 (0X08FD)

<u>Description</u>	WINSHOWCURSOR PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINSHOWCURSOR

**Minor Code**

2301 (0X08FD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FSHOW = %W, HWND = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2302 (0X08FE)

**Description**

WINCREATECURSOR PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCREATECURSOR

**Minor Code**

2302 (0X08FE)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RCLCLIP = %A, FS = %W, CY = %W

CX = %W, Y = %W, X = %W, HWND = %A

%RCLCLIP-&gt;XLEFT = %D, PRCLCLIP-&gt;YBOTTOM = %D

PRCLCLIP-&gt;XRIGHT = %D, PRCLCLIP-&gt;YTOP = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2303 (0X08FF)

**Description**

WINQUERYCURSORINFO PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCURSORINFO

**Minor Code**

2303 (0X08FF)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CURSORINFO = %A, HWNDDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2304 (0X0900)

**Description**

WINSETPOINTER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSETPOINTER

**Minor Code**

2304 (0X0900)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HPTRNEW = %A, HWNDDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2305 (0X0901)

**Description**

WINSHOWPOINTER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINSHOWPOINTER

**Minor Code**

2305 (0X0901)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%FSHOW = %W, HWNDDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2306 (0X0902)

<u>Description</u>	WINQUERYSYSPOINTER PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYSYSPOINTER
<u>Minor Code</u>	2306 (0X0902)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%FLOAD = %W, IPTR = %W, HWNDDESKTOP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2307 (0X0903)

<u>Description</u>	WINLOADPOINTER PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINLOADPOINTER
<u>Minor Code</u>	2307 (0X0903)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%IDRES = %W, HMOD = %W, HWNDDESKTOP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2308 (0X0904)

<u>Description</u>	WINCREATEPOINTER PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINCREATEPOINTER
<u>Minor Code</u>	2308 (0X0904)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.

**Traced Parameters**

%YHOTSPOT = %W, XHOTSPOT = %W, FPOINTER = %W  
HBMPOINTER = %A, HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2309 (0X0905)

**Description**

WINDESTROYPOINTER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYPOINTER

**Minor Code**

2309 (0X0905)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HPTR = %A

PMWIN Major Code: 0X00C2 Minor Code: 2310 (0X0906)

**Description**

WINQUERYPOINTER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYPOINTER

**Minor Code**

2310 (0X0906)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HWNDDESKTOP = %A

PMWIN Major Code: 0X00C2 Minor Code: 2311 (0X0907)

<b><u>Description</u></b>	WINSETPOINTERPOS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETPOINTERPOS
<b><u>Minor Code</u></b>	2311 (0X0907)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Y = %W, X = %W, HWNDDESKTOP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2312 (0X0908)

<b><u>Description</u></b>	WINQUERYPOINTERPOS PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYPOINTERPOS
<b><u>Minor Code</u></b>	2312 (0X0908)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%TL = %A, HWNDDESKTOP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2313 (0X0909)

<b><u>Description</u></b>	WINQUERYPOINTERINFO PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYPOINTERINFO
<b><u>Minor Code</u></b>	2313 (0X0909)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %OINTERINFO = %A, HPTR = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2314 (0X090A)

**Description**                      WINDRAWPOINTER PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINDRAWPOINTER

**Minor Code**                      2314 (0X090A)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %FS = %W, HPTR = %A, Y = %W, X = %W, HPS = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2315 (0X090B)

**Description**                      WINGETSYSBITMAP PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.WINGETSYSBITMAP

**Minor Code**                      2315 (0X090B)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %IBM = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2400 (0X0960)



<u>Description</u>	WINDESTROYCURSOR POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYCURSOR
<u>Minor Code</u>	2400 (0X0960)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 2401 (0X0961)

<u>Description</u>	WINSHOWCURSOR POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSHOWCURSOR
<u>Minor Code</u>	2401 (0X0961)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 2402 (0X0962)

<u>Description</u>	WINCREATECURSOR POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATECURSOR
<u>Minor Code</u>	2402 (0X0962)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2403 (0X0963)

**Description**

WINQUERYCURSORINFO POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYCURSORINFO

**Minor Code**

2403 (0X0963)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2404 (0X0964)

**Description**

WINSETPOINTER POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETPOINTER

**Minor Code**

2404 (0X0964)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2405 (0X0965)

<u>Description</u>	WINSHOWPOINTER POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSHOWPOINTER
<u>Minor Code</u>	2405 (0X0965)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2406 (0X0966)

<u>Description</u>	WINQUERYSYSPOINTER POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYSYSPOINTER
<u>Minor Code</u>	2406 (0X0966)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2407 (0X0967)

<u>Description</u>	WINLOADPOINTER POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINLOADPOINTER
<u>Minor Code</u>	2407 (0X0967)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2408 (0X0968)

**Description**

WINCREATEPOINTER POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINCREATEPOINTER

**Minor Code**

2408 (0X0968)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2409 (0X0969)

**Description**

WINDESTROYPOINTER POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINDESTROYPOINTER

**Minor Code**

2409 (0X0969)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2410 (0X096A)

<b><u>Description</u></b>	WINQUERYPOINTER POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYPOINTER
<b><u>Minor Code</u></b>	2410 (0X096A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2411 (0X096B)

<b><u>Description</u></b>	WINSETPOINTERPOS POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINSETPOINTERPOS
<b><u>Minor Code</u></b>	2411 (0X096B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2412 (0X096C)

<b><u>Description</u></b>	WINQUERYPOINTERPOS POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYPOINTERPOS
<b><u>Minor Code</u></b>	2412 (0X096C)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2413 (0X096D)

**Description**

WINQUERYPOINTERINFO POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINQUERYPOINTERINFO

**Minor Code**

2413 (0X096D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2414 (0X096E)

**Description**

WINDRAWPOINTER POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINDRAWPOINTER

**Minor Code**

2414 (0X096E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2415 (0X096F)

<b><u>Description</u></b>	WINGETSYSBITMAP POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.POSTWINGETSYSBITMAP
<b><u>Minor Code</u></b>	2415 (0X096F)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2500 (0X09C4)

<b><u>Description</u></b>	WINSETHOOK PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETHOOK
<b><u>Minor Code</u></b>	2500 (0X09C4)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HMOD = %W, PFNHOOK = %A, IHOOK = %W, HMQ = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2501 (0X09C5)

<b><u>Description</u></b>	WINRELEASEHOOK PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINRELEASEHOOK
<b><u>Minor Code</u></b>	2501 (0X09C5)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HMOD = %W, PFNHOOK = %A, IHOOK = %W, HMQ = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2502 (0X09C6)

**Description**

WINCALLMSGFILTER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCALLMSGFILTER

**Minor Code**

2502 (0X09C6)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MSGF = %W, PQMSG = %A

%QMSG->HWND = %A, PQMSG->MSG = %W

PQMSG->MP1 = %D, PQMSG->MP2 = %D

PQMSG->TIME = %D, PQMSG->PTL.X = %D

PQMSG->PTL.Y = %D

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2552 (0X09F8)

**Description**

FARCALLHOOK PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.FARCALLHOOK

**Minor Code**

2552 (0X09F8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**



%IHOOK = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2553 (0X09F9)

### Description

FREEQUEUEWINDOWHOOKS PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.FREEQUEUEWINDOWHOOKS

### Minor Code

2553 (0X09F9)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%FBADEXIT = %W, SMQ = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2600 (0X0A28)

### Description

WINSETHOOK POST-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WinSetHook

### Minor Code

2600 (0X0A28)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2601 (0X0A29)

### Description

WINRELEASEHOOK POST-INVOCATION

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinReleaseHook
<b><u>Minor Code</u></b>	2601 (0X0A29)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 2602 (0X0A2A)

<b><u>Description</u></b>	WINCALLMSGFILTER POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinCallMsgFilter
<b><u>Minor Code</u></b>	2602 (0X0A2A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

## PMWIN Major Code: 0X00C2 Minor Code: 2700 (0X0A8C)

<b><u>Description</u></b>	WINSETCP PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINSETCP
<b><u>Minor Code</u></b>	2700 (0X0A8C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

%IDCODEPAGE = %W, HMQ = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2701 (0X0A8D)

### Description

WINQUERYCP PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCP

### Minor Code

2701 (0X0A8D)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%HMQ = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2702 (0X0A8E)

### Description

WINQUERYCPLIST PRE-INVOCATION

### Tracepoint

Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCPLIST

### Minor Code

2702 (0X0A8E)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%RGCP = %A, CCPMAX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2703 (0X0A8F)

### Description

WINCPTRANSLATESTRING PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCPTRANSLATESTRING

**Minor Code**

2703 (0X0A8F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CHDEST = %A, CCHDESTMAX = %W

CPDST = %W, PSZSRC = %A, CPSRC = %W

PSZSRC -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2704 (0X0A90)

**Description**

WINCPTRANSLATECHAR PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINCPTRANSLATECHAR

**Minor Code**

2704 (0X0A90)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CPDST = %W, CHSRC = %B, CPSRC = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2705 (0X0A91)

**Description**

WINUPPER PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINUPPER

**Minor Code**

2705 (0X0A91)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%SZ = %A, IDCC = %W, IDCP = %W

PSZ-> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2706 (0X0A92)

**Description**

WINUPPERCHAR PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINUPPERCHAR

**Minor Code**

2706 (0X0A92)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%C = %W, IDCC = %W, IDCP = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2707 (0X0A93)

**Description**

WINNEXTCHAR PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINNEXTCHAR

**Minor Code**

2707 (0X0A93)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%SZ = %A, IDCC = %W, IDCP = %W

PSZ -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2708 (0X0A94)

<u>Description</u>	WINPREVCHAR PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINPREVCHAR
<u>Minor Code</u>	2708 (0X0A94)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %SZ = %A, PSZSTART = %A, IDCC = %W, IDCP = %W PSZ -> %S PSZSTART -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2709 (0X0A95)

<u>Description</u>	WINCOMPARESTRINGS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINCOMPARESTRINGS
<u>Minor Code</u>	2709 (0X0A95)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %RESERVED = %W, PSZ2 = %A, PSZ1 = %A IDCC = %W, IDCP = %W PSZ2 -> %S PSZ1 -> %S

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2710 (0X0A96)

<u>Description</u>	WINLOADSTRING PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINLOADSTRING
<u>Minor Code</u>	2710 (0X0A96)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%CHBUFFER = %A, CCHMAX = %W, ID = %W, HMOD = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2711 (0X0A97)

<u>Description</u>	WINLOADMESSAGE PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINLOADMESSAGE
<u>Minor Code</u>	2711 (0X0A97)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%CHBUFFER = %A, CCHMAX = %W, ID = %W, HMOD = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2750 (0X0ABE)

<u>Description</u>	WINLOADCHARXLATETBL PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINLOADCHARXLATETBL
<u>Minor Code</u>	2750 (0X0ABE)
<u>Trace Groups</u>	No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
%IDCODEPAGE = %W, HMOD = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 2751 (0X0ABF)

Description  
WINSETCHARXLATETBL PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINSETCHARXLATETBL

Minor Code  
2751 (0X0ABF)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
%LPXLATETBL = %A, HMQ = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2752 (0X0AC0)

Description  
WINQUERYCHARXLATETBL PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYCHARXLATETBL

Minor Code  
2752 (0X0AC0)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
%HMQ = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2753 (0X0AC1)



<u>Description</u>	WINLOADVKEYGLYPHXLATETBL PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINLOADVKEYGLYPHXLATETBL
<u>Minor Code</u>	2753 (0X0AC1)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%IDOUTPUT = %W, IDLANGUAGE = %W, IDKBDTYPE = %W, HMOD = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2754 (0X0AC2)

<u>Description</u>	WINSETVKEYGLYPHXLATETBL PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINSETVKEYGLYPHXLATETBL
<u>Minor Code</u>	2754 (0X0AC2)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%IDOUTPUT = %W, LPXLATETBL = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2755 (0X0AC3)

<u>Description</u>	WINQUERYVKEYGLYPHXLATETBL PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINQUERYVKEYGLYPHXLATETBL
<u>Minor Code</u>	2755 (0X0AC3)
<u>Trace Groups</u>	No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
%IDOUTPUT = %W

-----

PMWIN Major Code: 0X00C2 Minor Code: 2756 (0X0AC4)

Description  
WINSETKBDLAYOUT PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINSETKBDLAYOUT

Minor Code  
2756 (0X0AC4)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
%IDKBDLAYOUT = %W, HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2757 (0X0AC5)

Description  
WINQUERYKBDLAYOUT PRE-INVOCATION

Tracepoint  
Public symbol defined dynamic tracepoint: PMWIN.WINQUERYKBDLAYOUT

Minor Code  
2757 (0X0AC5)

Trace Groups  
No groups assigned.

Trace Types  
No types assigned.

Traced Parameters  
  
%HWNDDESKTOP = %A

-----

PMWIN Major Code: 0X00C2 Minor Code: 2800 (0X0AF0)

<u>Description</u>	WINSETCP POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinSetCp
<u>Minor Code</u>	2800 (0X0AF0)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2801 (0X0AF1)

<u>Description</u>	WINQUERYCP POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinQueryCp
<u>Minor Code</u>	2801 (0X0AF1)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2802 (0X0AF2)

<u>Description</u>	WINQUERYCPLIST POST-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WinQueryCpList
<u>Minor Code</u>	2802 (0X0AF2)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2803 (0X0AF3)

**Description**

WINCPTRANSLATESTRING POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinCpTranslateString

**Minor Code**

2803 (0X0AF3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2804 (0X0AF4)

**Description**

WINCPTRANSLATECHAR POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinCpTranslateChar

**Minor Code**

2804 (0X0AF4)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2805 (0X0AF5)

<b><u>Description</u></b>	WINUPPER POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinUpper
<b><u>Minor Code</u></b>	2805 (0X0AF5)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2806 (0X0AF6)

<b><u>Description</u></b>	WINUPPERCHAR POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinUpperChar
<b><u>Minor Code</u></b>	2806 (0X0AF6)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2807 (0X0AF7)

<b><u>Description</u></b>	WINNEXTCHAR POST-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WinNextChar
<b><u>Minor Code</u></b>	2807 (0X0AF7)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2808 (0X0AF8)

**Description**

WINPREVCHAR POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WinPrevChar

**Minor Code**

2808 (0X0AF8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2809 (0X0AF9)

**Description**

WINCOMPARESTRINGS POST-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.POSTWINCOMPARESTRINGS

**Minor Code**

2809 (0X0AF9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DX = %W, AX = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2900 (0X0B54)

<b><u>Description</u></b>	WINCREATEHEAP PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINCREATEHEAP
<b><u>Minor Code</u></b>	2900 (0X0B54)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%FOPTIONS = %W, CBMAXDED = %W, CHMINDED = %W CBGROW = %W, CBHEAP = %W, SELHEAPBASE = %W

## PMWIN Major Code: 0X00C2 Minor Code: 2901 (0X0B55)

<b><u>Description</u></b>	WINDESTROYHEAP PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINDESTROYHEAP
<b><u>Minor Code</u></b>	2901 (0X0B55)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HHEAP = %A

## PMWIN Major Code: 0X00C2 Minor Code: 2902 (0X0B56)

<b><u>Description</u></b>	WINAVAILMEM PRE-INVOCATION
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMWIN.WINAVAILMEM
<b><u>Minor Code</u></b>	2902 (0X0B56)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CBMINFREE = %W, FCOMPACT = %W, HHEAP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2903 (0X0B57)

**Description**

WINALLOCMEM PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINALLOCMEM

**Minor Code**

2903 (0X0B57)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CB = %W, HHEAP = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2904 (0X0B58)

**Description**

WINREALLOCMEM PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.WINREALLOCMEM

**Minor Code**

2904 (0X0B58)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%CBNEW = %W, CBOLD = %W, NPMEM = %W, HHEAP = %A

-----



# PMWIN Major Code: 0X00C2 Minor Code: 2905 (0X0B59)

<u>Description</u>	WINFREEMEM PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINFREEMEM
<u>Minor Code</u>	2905 (0X0B59)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%CBMEM = %W, NPMEM = %W, HHEAP = %A

-----

# PMWIN Major Code: 0X00C2 Minor Code: 2906 (0X0B5A)

<u>Description</u>	WINLOCKHEAP PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.WINLOCKHEAP
<u>Minor Code</u>	2906 (0X0B5A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%HHEAP = %A

-----

# PMWIN Major Code: 0X00C2 Minor Code: 2950 (0X0B86)

<u>Description</u>	COMPACTMOVEABLEHEAP PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.COMPACTMOVEABLEHEAP
<u>Minor Code</u>	2950 (0X0B86)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
PHEAPCB = %A

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2951 (0X0B87)

**Description**                      FINDFREEBLOCK PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.FINDFREEBLOCK

**Minor Code**                      2951 (0X0B87)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
PHEAPCB = %A, CBBLOCKSIZE = %W

-----

## PMWIN Major Code: 0X00C2 Minor Code: 2952 (0X0B88)

**Description**                      FINDMAXFREEBLOCK PRE-INVOCATION

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMWIN.FINDMAXFREEBLOCK

**Minor Code**                      2952 (0X0B88)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**  
  
PHEAPCB = %A

# PMWIN Major Code: 0X00C2 Minor Code: 2953 (0X0B89)

<u>Description</u>	INSERTFREEBLOCK PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.INSERTFREEBLOCK
<u>Minor Code</u>	2953 (0X0B89)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	PHEAPCB = %A, PBLOCK = %W, CBBLOCKSIZE = %W

# PMWIN Major Code: 0X00C2 Minor Code: 2954 (0X0B8A)

<u>Description</u>	SORTFREELIST PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.SORTFREELIST
<u>Minor Code</u>	2954 (0X0B8A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	PHEAPCB = %A, PFREELISTHEAD = %A, CBBLOCKSIZE = %W

# PMWIN Major Code: 0X00C2 Minor Code: 2955 (0X0B8B)

<u>Description</u>	GETSIZEDS PRE-INVOCATION
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMWIN.GETSIZEDS
<u>Minor Code</u>	

2955 (0X0B8B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

DS = %W

---

## PMWIN Major Code: 0X00C2 Minor Code: 2956 (0X0B8C)

**Description**

VALIDATEHEAPHANDLE PRE-INVOCATION

**Tracepoint**

Public symbol defined dynamic tracepoint: PMWIN.VALIDATEHEAPHANDLE

**Minor Code**

2956 (0X0B8C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

No parameters traced.

---

## PMGRE.DLL Trace Events

The tracepoints for the PMGRE.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for PMGRE Major Code: 0X00C3, sorted by minor code.](#)

[Trace events for PMGRE Major Code: 0X00C3 ,sorted by tracepoint.](#)

---

## Trace Events for PMGRE Major Code: 0X00C3, Sorted by Minor Code

00002 (0X0002) GREENTRY2 Pre-Invocation  
00003 (0X0003) GREENTRY3 Pre-Invocation  
00004 (0X0004) GREENTRY4 Pre-Invocation

00005 (0X0005) GREENTRY5 Pre-Invocation  
00006 (0X0006) GREENTRY6 Pre-Invocation  
00007 (0X0007) GREENTRY7 Pre-Invocation  
00008 (0X0008) GREENTRY8 Pre-Invocation  
00009 (0X0009) GREENTRY9 Pre-Invocation  
00010 (0X000A) GREENTRY10 Pre-Invocation  
00012 (0X000C) GRE32ENTRY2 Pre-Invocation  
00013 (0X000D) GRE32ENTRY3 Pre-Invocation  
00014 (0X000E) GRE32ENTRY4 Pre-Invocation  
00015 (0X000F) GRE32ENTRY5 Pre-Invocation  
00016 (0X0010) GRE32ENTRY6 Pre-Invocation  
00017 (0X0011) GRE32ENTRY7 Pre-Invocation  
00018 (0X0012) GRE32ENTRY8 Pre-Invocation  
00019 (0X0013) GRE32ENTRY9 Pre-Invocation  
00020 (0X0014) GRE32ENTRY10 Pre-Invocation  
00022 (0X0016) INNERGRE32ENTRY2 Pre-Invocation  
00023 (0X0017) INNERGRE32ENTRY3 Pre-Invocation  
00024 (0X0018) INNERGRE32ENTRY4 Pre-Invocation  
00025 (0X0019) INNERGRE32ENTRY5 Pre-Invocation  
00026 (0X001A) INNERGRE32ENTRY6 Pre-Invocation  
00027 (0X001B) INNERGRE32ENTRY7 Pre-Invocation  
00028 (0X001C) INNERGRE32ENTRY8 Pre-Invocation  
00029 (0X001D) INNERGRE32ENTRY9 Pre-Invocation  
00030 (0X001E) INNERGRE32ENTRY10 Pre-Invocation

---

## Trace Events for PMGRE Major Code: 0X00C3, Sorted by Tracepoint

GRE32ENTRY10 00020 (0X0014)  
GRE32ENTRY2 00012 (0X000C)  
GRE32ENTRY3 00013 (0X000D)  
GRE32ENTRY4 00014 (0X000E)  
GRE32ENTRY5 00015 (0X000F)  
GRE32ENTRY6 00016 (0X0010)  
GRE32ENTRY7 00017 (0X0011)  
GRE32ENTRY8 00018 (0X0012)  
GRE32ENTRY9 00019 (0X0013)  
GREENTRY10 00010 (0X000A)  
GREENTRY2 00002 (0X0002)  
GREENTRY3 00003 (0X0003)  
GREENTRY4 00004 (0X0004)  
GREENTRY5 00005 (0X0005)  
GREENTRY6 00006 (0X0006)  
GREENTRY7 00007 (0X0007)  
GREENTRY8 00008 (0X0008)  
GREENTRY9 00009 (0X0009)  
INNERGRE32ENTRY10 00030 (0X001E)  
INNERGRE32ENTRY2 00022 (0X0016)  
INNERGRE32ENTRY3 00023 (0X0017)  
INNERGRE32ENTRY4 00024 (0X0018)  
INNERGRE32ENTRY5 00025 (0X0019)  
INNERGRE32ENTRY6 00026 (0X001A)  
INNERGRE32ENTRY7 00027 (0X001B)  
INNERGRE32ENTRY8 00028 (0X001C)  
INNERGRE32ENTRY9 00029 (0X001D)

---

PMGRE Major Code: 0X00C3 Minor Code: 2 (0X0002)

<u>Description</u>	GREENTRY2 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.GREENTRY2
<u>Minor Code</u>	2 (0X0002)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Return Address = %a, lparam2 = %d, lparam1 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 3 (0X0003)

<u>Description</u>	GREENTRY3 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.GREENTRY3
<u>Minor Code</u>	3 (0X0003)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Return Address = %a, lparam3 = %d, lparam2 = %d lparam1 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 4 (0X0004)

<u>Description</u>	GREENTRY4 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.GREENTRY4
<u>Minor Code</u>	4 (0X0004)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Return Address = %a, lparam4 = %d, lparam3 = %d  
lparam2 = %d, lparam1 = %d

-----

PMGRE Major Code: 0X00C3 Minor Code: 5 (0X0005)

**Description**

GREENTRY5 Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGRE.GREENTRY5

**Minor Code**

5 (0X0005)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Return Address = %a, lparam5 = %d, lparam4 = %d  
lparam3 = %d, lparam2 = %d, lparam1 = %d

-----

PMGRE Major Code: 0X00C3 Minor Code: 6 (0X0006)

**Description**

GREENTRY6 Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGRE.GREENTRY6

**Minor Code**

6 (0X0006)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Return Address = %a, lparam6 = %d, lparam5 = %d  
lparam4 = %d, lparam3 = %d, lparam2 = %d

lparam1 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 7 (0X0007)

### Description

GREENTRY7 Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGRE.GREENTRY7

### Minor Code

7 (0X0007)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Return Address = %a, lparam7 = %d, lparam6 = %d

lparam5 = %d, lparam4 = %d, lparam3 = %d,

lparam2 = %d ,lparam1 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 8 (0X0008)

### Description

GREENTRY8 Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGRE.GREENTRY8

### Minor Code

8 (0X0008)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Return Address = %a, lparam8 = %d, lparam7 = %d

lparam6 = %d, lparam5 = %d, lparam4 = %d,

lparam3 = %d, lparam2 = %d, lparam1 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 9 (0X0009)



<b><u>Description</u></b>	GREENTRY9 Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGRE.GREENTRY9
<b><u>Minor Code</u></b>	9 (0X0009)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

%Return Address = %a, lparam9 = %d, lparam8 = %d  
lparam7 = %d, lparam6 = %d, lparam5 = %d  
lparam4 = %d, lparam3 = %d, lparam2 = %d  
lparam1 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 10 (0X000A)

<b><u>Description</u></b>	GREENTRY10 Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGRE.GREENTRY10
<b><u>Minor Code</u></b>	10 (0X000A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

%Return Address = %a, lparam10 = %d, lparam9 = %d  
lparam8 = %d, lparam7 = %d, lparam6 = %d,  
lparam5 = %d, lparam4 = %d, lparam3 = %d,  
lparam2 = %d, lparam1 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 12 (0X000C)

<u>Description</u>	GRE32ENTRY2 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY2
<u>Minor Code</u>	12 (0X000C)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Return Address = %f, lparam1 = %d, lparam2 = %d

-----

PMGRE Major Code: 0X00C3 Minor Code: 13 (0X000D)

<u>Description</u>	GRE32ENTRY3 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY3
<u>Minor Code</u>	13 (0X000D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Return Address = %f, lparam1 = %d, lparam2 = %d lparam3 = %d

-----

PMGRE Major Code: 0X00C3 Minor Code: 14 (0X000E)

<u>Description</u>	GRE32ENTRY4 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY4
<u>Minor Code</u>	14 (0X000E)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Return Address = %f, lparam1 = %d, lparam2 = %d  
lparam3 = %d, lparam4 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 15 (0X000F)

**Description**  
GRE32ENTRY5 Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY5

**Minor Code**  
15 (0X000F)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Return Address = %f, lparam1 = %d, lparam2 = %d  
lparam3 = %d, lparam4 = %d, lparam5 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 16 (0X0010)

**Description**  
GRE32ENTRY6 Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY6

**Minor Code**  
16 (0X0010)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Return Address = %f, lparam1 = %d, lparam2 = %d  
lparam3 = %d, lparam4 = %d, lparam5 = %d

lparam6 = %d

---

## PMGRE Major Code: 0X00C3 Minor Code: 17 (0X0011)

### Description

GRE32ENTRY7 Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY7

### Minor Code

17 (0X0011)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d

lparam3 = %d, lparam4 = %d, lparam5 = %d

lparam6 = %d ,lparam7 = %d

---

## PMGRE Major Code: 0X00C3 Minor Code: 18 (0X0012)

### Description

GRE32ENTRY8 Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY8

### Minor Code

18 (0X0012)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d

lparam3 = %d, lparam4 = %d, lparam5 = %d

lparam6 = %d, lparam7 = %d, lparam8 = %d

---

## PMGRE Major Code: 0X00C3 Minor Code: 19 (0X0013)

<b><u>Description</u></b>	GRE32ENTRY9 Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY9
<b><u>Minor Code</u></b>	19 (0X0013)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Return Address = %f, lparam1 = %d, lparam2 = %d lparam3 = %d, lparam4 = %d, lparam5 = %d lparam6 = %d, lparam7 = %d, lparam8 = %d lparam9 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 20 (0X0014)

<b><u>Description</u></b>	GRE32ENTRY10 Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGRE.GRE32ENTRY10
<b><u>Minor Code</u></b>	20 (0X0014)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Return Address = %f, lparam1 = %d, lparam2 = %d lparam3 = %d, lparam4 = %d, lparam5 = %d lparam6 = %d, lparam7 = %d, lparam8 = %d lparam9 = %d, lparam10 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 22 (0X0016)

<u>Description</u>	INNERGRE32ENTRY2 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY2
<u>Minor Code</u>	22 (0X0016)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Return Address = %f, lparam1 = %d, lparam2 = %d

-----

PMGRE Major Code: 0X00C3 Minor Code: 23 (0X0017)

<u>Description</u>	INNERGRE32ENTRY3 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY3
<u>Minor Code</u>	23 (0X0017)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Return Address = %f, lparam1 = %d, lparam2 = %d lparam3 = %d

-----

PMGRE Major Code: 0X00C3 Minor Code: 24 (0X0018)

<u>Description</u>	INNERGRE32ENTRY4 Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY4
<u>Minor Code</u>	24 (0X0018)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Return Address = %f, lparam1 = %d, lparam2 = %d

lparam3 = %d, lparam4 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 25 (0X0019)

**Description**

INNERGRE32ENTRY5 Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY5

**Minor Code**

25 (0X0019)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Return Address = %f, lparam1 = %d, lparam2 = %d

lparam3 = %d, lparam4 = %d, lparam5 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 26 (0X001A)

**Description**

INNERGRE32ENTRY6 Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY6

**Minor Code**

26 (0X001A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Return Address = %f, lparam1 = %d, lparam2 = %d

lparam3 = %d, lparam4 = %d, lparam5 = %d

lparam6 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 27 (0X001B)

### Description

INNERGRE32ENTRY7 Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY7

### Minor Code

27 (0X001B)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d

lparam3 = %d, lparam4 = %d, lparam5 = %d

lparam6 = %d ,lparam7 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 28 (0X001C)

### Description

INNERGRE32ENTRY8 Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY8

### Minor Code

28 (0X001C)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Return Address = %f, lparam1 = %d, lparam2 = %d

lparam3 = %d, lparam4 = %d, lparam5 = %d

lparam6 = %d, lparam7 = %d, lparam8 = %d

-----

## PMGRE Major Code: 0X00C3 Minor Code: 29 (0X001D)



<b>Description</b>	INNERGRE32ENTRY9 Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY9
<b>Minor Code</b>	29 (0X001D)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	%Return Address = %f, lparam1 = %d, lparam2 = %d lparam3 = %d, lparam4 = %d, lparam5 = %d lparam6 = %d, lparam7 = %d, lparam8 = %d lparam9 = %d

-----

PMGRE Major Code: 0X00C3 Minor Code: 30 (0X001E)

<b>Description</b>	INNERGRE32ENTRY10 Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMGRE.INNERGRE32ENTRY10
<b>Minor Code</b>	30 (0X001E)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	%Return Address = %f, lparam1 = %d, lparam2 = %d lparam3 = %d, lparam4 = %d, lparam5 = %d lparam6 = %d, lparam7 = %d, lparam8 = %d

-----

PMPIC.DLL Trace Events

The tracepoints for the PMPIC.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

Trace events for PMPIC Major Code: 0X00C4, sorted by minor code.  
Trace events for PMPIC Major Code: 0X00C4 ,sorted by tracepoint.

Trace Events for PMPIC Major Code: 0X00C4, Sorted by Minor Code

00010 (0X000A) Piclchg Pre-Invocation  
01000 (0X03E8) PicPrint Pre-Invocation

Trace Events for PMPIC Major Code: 0X00C4, Sorted by Tracepoint

PICICHG 00010 (0X000A)  
PICPRINT 01000 (0X03E8)

PMPIC Major Code: 0X00C4 Minor Code: 10 (0X000A)

Description	Piclchg Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMPIC.PICICHG
Minor Code	10 (0X000A)
Trace Groups	No groups assigned.
Trace Types	No types assigned.
Traced Parameters	%convtype = %d, out_file = %d, full_file = %d, HabPic = %d

PMPIC Major Code: 0X00C4 Minor Code: 1000 (0X03E8)

<b>Description</b>	PicPrint Pre-Invocation
<b>Tracepoint</b>	Public symbol defined dynamic tracepoint: PMPIC.PICPRINT
<b>Minor Code</b>	1000 (0X03E8)
<b>Trace Groups</b>	No groups assigned.
<b>Trace Types</b>	No types assigned.
<b>Traced Parameters</b>	%aram_str = %a, type = %d, filename = %a, hab = %d

## PMGPI.DLL Trace Events

The tracepoints for the PMGPI.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

Delay:

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for PMGPI Major Code: 0X00C5, sorted by minor code.](#)  
[Trace events for PMGPI Major Code: 0X00C5 ,sorted by tracepoint.](#)

## Trace Events for PMGPI Major Code: 0X00C5, Sorted by Minor Code

00101 (0X0065) GpiCreatePS Pre-Invocation  
00102 (0X0066) GpiDestroyPS Pre-Invocation  
00103 (0X0067) GpiAssociate Pre-Invocation  
00104 (0X0068) GpiRestorePS Pre-Invocation  
00105 (0X0069) GpiSavePS Pre-Invocation  
00106 (0X006A) GpiErase Pre-Invocation  
00107 (0X006B) GpiQueryDevice Pre-Invocation  
00108 (0X006C) GpiResetPS Pre-Invocation  
00109 (0X006D) GpiSetPS Pre-Invocation  
00110 (0X006E) GpiQueryPS Pre-Invocation  
00111 (0X006F) GpiErrorSegmentData Pre-Invocation  
00112 (0X0070) GpiQueryDrawControl Pre-Invocation  
00113 (0X0071) GpiSetDrawControl Pre-Invocation  
00114 (0X0072) GpiQueryDrawingMode Pre-Invocation  
00115 (0X0073) GpiSetDrawingMode Pre-Invocation  
00116 (0X0074) GpiQueryStopDraw Pre-Invocation  
00117 (0X0075) GpiSetStopDraw Pre-Invocation  
00201 (0X00C9) GpiCorrelateChain Pre-Invocation  
00202 (0X00CA) GpiQueryTag Pre-Invocation  
00203 (0X00CB) GpiSetTag Pre-Invocation

00204 (0X00CC) GpiQueryPickApertureSize Pre-Invocation  
00205 (0X00CD) GpiSetPickApertureSize Pre-Invocation  
00206 (0X00CE) GpiQueryPickAperturePosition Pre-Invocation  
00207 (0X00CF) GpiSetPickAperturePosition Pre-Invocation  
00208 (0X00D0) GpiQueryBoundaryData Pre-Invocation  
00209 (0X00D1) GpiResetBoundaryData Pre-Invocation  
00210 (0X00D2) GpiCorrelateFrom Pre-Invocation  
00211 (0X00D3) GpiCorrelateSegment Pre-Invocation  
00300 (0X012C) GpiOpenSegment Pre-Invocation  
00301 (0X012D) GpiCloseSegment Pre-Invocation  
00302 (0X012E) GpiDeleteSegment Pre-Invocation  
00303 (0X012F) GpiQueryInitialSegmentAttrs Pre-Invocation  
00304 (0X0130) GpiSetInitialSegmentAttrs Pre-Invocation  
00305 (0X0131) GpiQuerySegmentAttrs Pre-Invocation  
00306 (0X0132) GpiSetSegmentAttrs Pre-Invocation  
00307 (0X0133) GpiQuerySegmentPriority Pre-Invocation  
00308 (0X0134) GpiSetSegmentPriority Pre-Invocation  
00309 (0X0135) GpiDeleteSegments Pre-Invocation  
00310 (0X0136) GpiQuerySegmentNames Pre-Invocation  
00400 (0X0190) GpiGetData Pre-Invocation  
00401 (0X0191) GpiPutData Pre-Invocation  
00402 (0X0192) GpiDrawChain Pre-Invocation  
00403 (0X0193) GpiDrawFrom Pre-Invocation  
00404 (0X0194) GpiDrawSegment Pre-Invocation  
00405 (0X0195) GpiDrawDynamics Pre-Invocation  
00406 (0X0196) GpiRemoveDynamics Pre-Invocation  
00500 (0X01F4) GpiBeginElement Pre-Invocation  
00501 (0X01F5) GpiEndElement Pre-Invocation  
00502 (0X01F6) GpiLabel Pre-Invocation  
00503 (0X01F7) GpiElement Pre-Invocation  
00504 (0X01F8) GpiQueryElement Pre-Invocation  
00505 (0X01F9) GpiDeleteElement Pre-Invocation  
00506 (0X01FA) GpiDeleteElementRange Pre-Invocation  
00507 (0X01FB) GpiDeleteElementsBetweenLabels Pre-Invocation  
00508 (0X01FC) GpiQueryEditMode Pre-Invocation  
00509 (0X01FD) GpiSetEditMode Pre-Invocation  
00510 (0X01FE) GpiQueryElementPointer Pre-Invocation  
00511 (0X01FF) GpiSetElementPointer Pre-Invocation  
00512 (0X0200) GpiOffsetElementPointer Pre-Invocation  
00513 (0X0201) GpiQueryElementType Pre-Invocation  
00514 (0X0202) GpiSetElementPointerAtLabel Pre-Invocation  
00600 (0X0258) GpiQuerySegmentTransformMatrix Pre-Invocation  
00601 (0X0259) GpiSetSegmentTransformMatrix Pre-Invocation  
00602 (0X025A) GpiConvert Pre-Invocation  
00603 (0X025B) GpiQueryModelTransformMatrix Pre-Invocation  
00604 (0X025C) GpiSetModelTransformMatrix Pre-Invocation  
00605 (0X025D) GpiCallSegmentMatrix Pre-Invocation  
00606 (0X025E) GpiQueryDefaultViewMatrix Pre-Invocation  
00607 (0X025F) GpiSetDefaultViewMatrix Pre-Invocation  
00608 (0X0260) GpiQueryPageViewport Pre-Invocation  
00609 (0X0261) GpiSetPageViewport Pre-Invocation  
00610 (0X0262) GpiQueryViewingTransformMatrix Pre-Invocation  
00611 (0X0263) GpiSetViewingTransformMatrix Pre-Invocation  
00700 (0X02BC) GpiSetGraphicsField Pre-Invocation  
00701 (0X02BD) GpiQueryGraphicsField Pre-Invocation  
00702 (0X02BE) GpiSetViewingLimits Pre-Invocation  
00703 (0X02BF) GpiQueryViewingLimits Pre-Invocation  
00800 (0X0320) GpiBeginPath Pre-Invocation  
00801 (0X0321) GpiEndPath Pre-Invocation  
00802 (0X0322) GpiCloseFigure Pre-Invocation  
00803 (0X0323) GpiModifyPath Pre-Invocation  
00804 (0X0324) GpiFillPath Pre-Invocation  
00805 (0X0325) GpiSetClipPath Pre-Invocation  
00806 (0X0326) GpiStrokePath Pre-Invocation  
00900 (0X0384) GpiCreateLogColorTable Pre-Invocation  
00901 (0X0385) GpiRealizeColorTable Pre-Invocation  
00902 (0X0386) GpiUnrealizeColorTable Pre-Invocation  
00903 (0X0387) GpiQueryColorData Pre-Invocation  
00904 (0X0388) GpiQueryLogColorTable Pre-Invocation  
00905 (0X0389) GpiQueryRealColors Pre-Invocation  
00906 (0X038A) GpiQueryNearestColor Pre-Invocation  
00907 (0X038B) GpiQueryColorIndex Pre-Invocation  
00908 (0X038C) GpiQueryRGBColor Pre-Invocation

01000 (0X03E8) GpiSetColor Pre-Invocation  
01001 (0X03E9) GpiQueryColor Pre-Invocation  
01002 (0X03EA) GpiSetBackColor Pre-Invocation  
01003 (0X03EB) GpiQueryBackColor Pre-Invocation  
01004 (0X03EC) GpiSetMix Pre-Invocation  
01005 (0X03ED) GpiQueryMix Pre-Invocation  
01006 (0X03EE) GpiSetBackMix Pre-Invocation  
01007 (0X03EF) GpiQueryBackMix Pre-Invocation  
01008 (0X03F0) GpiSetAttrs Pre-Invocation  
01009 (0X03F1) GpiQueryAttrs Pre-Invocation  
01010 (0X03F2) GpiSetAttrMode Pre-Invocation  
01011 (0X03F3) GpiQueryAttrMode Pre-Invocation  
01100 (0X044C) GpiSetPatternSet Pre-Invocation  
01101 (0X044D) GpiQueryPatternSet Pre-Invocation  
01102 (0X044E) GpiSetPatternRefPoint Pre-Invocation  
01103 (0X044F) GpiQueryPatternRefPoint Pre-Invocation  
01104 (0X0450) GpiSetPattern Pre-Invocation  
01105 (0X0451) GpiQueryPattern Pre-Invocation  
01106 (0X0452) GpiBeginArea Pre-Invocation  
01107 (0X0453) GpiEndArea Pre-Invocation  
01200 (0X04B0) GpiSetLineType Pre-Invocation  
01201 (0X04B1) GpiQueryLineType Pre-Invocation  
01202 (0X04B2) GpiSetLineWidth Pre-Invocation  
01203 (0X04B3) GpiQueryLineWidth Pre-Invocation  
01204 (0X04B4) GpiSetLineWidthGeom Pre-Invocation  
01205 (0X04B5) GpiQueryLineWidthGeom Pre-Invocation  
01206 (0X04B6) GpiSetLineEnd Pre-Invocation  
01207 (0X04B7) GpiQueryLineEnd Pre-Invocation  
01208 (0X04B8) GpiSetLineJoin Pre-Invocation  
01209 (0X04B9) GpiQueryLineJoin Pre-Invocation  
01210 (0X04BA) GpiSetCurrentPosition Pre-Invocation  
01211 (0X04BB) GpiQueryCurrentPosition Pre-Invocation  
01212 (0X04BC) GpiBox Pre-Invocation  
01213 (0X04BD) GpiMove Pre-Invocation  
01214 (0X04BE) GpiLine Pre-Invocation  
01215 (0X04BF) GpiPolyLine Pre-Invocation  
01300 (0X0514) GpiSetArcParams Pre-Invocation  
01301 (0X0515) GpiQueryArcParams Pre-Invocation  
01302 (0X0516) GpiPointArc Pre-Invocation  
01303 (0X0517) GpiFullArc Pre-Invocation  
01304 (0X0518) GpiPartialArc Pre-Invocation  
01305 (0X0519) GpiPolyFilletSharp Pre-Invocation  
01306 (0X051A) GpiPolySpline Pre-Invocation  
01307 (0X051B) GpiPolyFillet Pre-Invocation  
01400 (0X0578) GpiQueryTextBox Pre-Invocation  
01401 (0X0579) GpiQueryDefCharBox Pre-Invocation  
01402 (0X057A) GpiSetCharSet Pre-Invocation  
01403 (0X057B) GpiQueryCharSet Pre-Invocation  
01404 (0X057C) GpiSetCharBox Pre-Invocation  
01405 (0X057D) GpiQueryCharBox Pre-Invocation  
01406 (0X057E) GpiSetCharAngle Pre-Invocation  
01407 (0X057F) GpiQueryCharAngle Pre-Invocation  
01408 (0X0580) GpiSetCharShear Pre-Invocation  
01409 (0X0581) GpiQueryCharShear Pre-Invocation  
01410 (0X0582) GpiSetCharDirection Pre-Invocation  
01411 (0X0583) GpiQueryCharDirection Pre-Invocation  
01412 (0X0584) GpiSetCharMode Pre-Invocation  
01413 (0X0585) GpiQueryCharMode Pre-Invocation  
01414 (0X0586) GpiCharStringPos Pre-Invocation  
01415 (0X0587) GpiCharStringPosAt Pre-Invocation  
01416 (0X0588) GpiCharString Pre-Invocation  
01417 (0X0589) GpiCharStringAt Pre-Invocation  
01418 (0X058A) GpiQueryCharStringPos Pre-Invocation  
01419 (0X058B) GpiQueryCharStringPosAt Pre-Invocation  
01500 (0X05DC) GpiSetMarkerSet Pre-Invocation  
01501 (0X05DD) GpiQueryMarkerSet Pre-Invocation  
01502 (0X05DE) GpiSetMarker Pre-Invocation  
01503 (0X05DF) GpiQueryMarker Pre-Invocation  
01504 (0X05E0) GpiSetMarkerBox Pre-Invocation  
01505 (0X05E1) GpiQueryMarkerBox Pre-Invocation  
01506 (0X05E2) GpiMarker Pre-Invocation  
01507 (0X05E3) GpiPolyMarker Pre-Invocation  
01600 (0X0640) GpiImage Pre-Invocation

01601 (0X0641) GpiPop Pre-Invocation  
01602 (0X0642) GpiPtVisible Pre-Invocation  
01603 (0X0643) GpiRectVisible Pre-Invocation  
01604 (0X0644) GpiComment Pre-Invocation  
01700 (0X06A4) GpiDeleteSetId Pre-Invocation  
01701 (0X06A5) GpiQueryNumberSetIds Pre-Invocation  
01702 (0X06A6) GpiQuerySetIds Pre-Invocation  
01703 (0X06A7) GpiLoadFonts Pre-Invocation  
01704 (0X06A8) GpiUnloadFonts Pre-Invocation  
01705 (0X06A9) GpiCreateLogFont Pre-Invocation  
01706 (0X06AA) GpiQueryFonts Pre-Invocation  
01707 (0X06AB) GpiQueryFontMetrics Pre-Invocation  
01708 (0X06AC) GpiQueryKerningPairs Pre-Invocation  
01709 (0X06AD) GpiQueryWidthTable Pre-Invocation  
01710 (0X06AE) GpiSetCp Pre-Invocation  
01711 (0X06AF) GpiQueryCp Pre-Invocation  
01712 (0X06B0) GpiQueryFontFileDescriptions Pre-Invocation  
01800 (0X0708) GpiDeleteBitmap Pre-Invocation  
01801 (0X0709) GpiSetBitmap Pre-Invocation  
01802 (0X070A) GpiBitBlt Pre-Invocation  
01803 (0X070B) GpiWCBitBlt Pre-Invocation  
01804 (0X070C) GpiCreateBitmap Pre-Invocation  
01805 (0X070D) GpiSetBitmapDimension Pre-Invocation  
01806 (0X070E) GpiQueryBitmapDimension Pre-Invocation  
01807 (0X070F) GpiQueryDeviceBitmapFormats Pre-Invocation  
01808 (0X0710) GpiQueryBitmapParameters Pre-Invocation  
01809 (0X0711) GpiQueryBitmapBits Pre-Invocation  
01810 (0X0712) GpiSetBitmapBits Pre-Invocation  
01811 (0X0713) GpiSetPel Pre-Invocation  
01812 (0X0714) GpiQueryPel Pre-Invocation  
01813 (0X0715) GpiSetBitmapId Pre-Invocation  
01814 (0X0716) GpiQueryBitmapHandle Pre-Invocation  
01900 (0X076C) GpiCreateRegion Pre-Invocation  
01901 (0X076D) GpiSetRegion Pre-Invocation  
01902 (0X076E) GpiDestroyRegion Pre-Invocation  
01903 (0X076F) GpiCombineRegion Pre-Invocation  
01904 (0X0770) GpiEqualRegion Pre-Invocation  
01905 (0X0771) GpiOffsetRegion Pre-Invocation  
01906 (0X0772) GpiPtInRegion Pre-Invocation  
01907 (0X0773) GpiRectInRegion Pre-Invocation  
01908 (0X0774) GpiQueryRegionBox Pre-Invocation  
01909 (0X0775) GpiQueryRegionRects Pre-Invocation  
01910 (0X0776) GpiPaintRegion Pre-Invocation  
02000 (0X07D0) GpiSetClipRegion Pre-Invocation  
02001 (0X07D1) GpiQueryClipRegion Pre-Invocation  
02002 (0X07D2) GpiQueryClipBox Pre-Invocation  
02003 (0X07D3) GpiIntersectClipRectangle Pre-Invocation  
02004 (0X07D4) GpiExcludeClipRectangle Pre-Invocation  
02005 (0X07D5) GpiOffsetClipRegion Pre-Invocation  
02100 (0X0834) GpiLoadMetaFile Pre-Invocation  
02101 (0X0835) GpiPlayMetaFile Pre-Invocation  
02102 (0X0836) GpiSaveMetaFile Pre-Invocation  
02103 (0X0837) GpiDeleteMetaFile Pre-Invocation  
02104 (0X0838) GpiCopyMetaFile Pre-Invocation  
02105 (0X0839) GpiQueryMetaFileLength Pre-Invocation  
02106 (0X083A) GpiQueryMetaFileBits Pre-Invocation  
02107 (0X083B) GpiSetMetaFileBits Pre-Invocation  
02300 (0X08FC) MTIDPStoreMeta Pre-Invocation  
02301 (0X08FD) MTEnableKerningMeta Pre-Invocation  
02302 (0X08FE) MTDisplayFlagMeta Pre-Invocation  
02303 (0X08FF) MTCreateLogColorTableMeta Pre-Invocation  
02304 (0X0900) MTSetCodePageMeta Pre-Invocation  
02305 (0X0901) MTDeleteSetIDMeta Pre-Invocation  
02306 (0X0902) MTSetGraphicsFieldMeta Pre-Invocation  
02307 (0X0903) MTResetMeta Pre-Invocation  
02308 (0X0904) MTEraseMeta Pre-Invocation  
02309 (0X0905) MTAssociateMeta Pre-Invocation  
02310 (0X0906) MTVerifyPageUnits Pre-Invocation  
02311 (0X0907) MTBitBltMeta Pre-Invocation  
02312 (0X0908) MTWCBitBltMeta Pre-Invocation  
02313 (0X0909) MTSetPelMeta Pre-Invocation  
02314 (0X090A) MTSelectClipMeta Pre-Invocation  
02315 (0X090B) MTClipRegionMeta Pre-Invocation

02316 (0X090C) MTPaintRegionMeta Pre-Invocation  
02317 (0X090D) MTDevEscape Pre-Invocation  
02318 (0X090E) MTRestorePS Pre-Invocation  
02319 (0X090F) MTSavePS Pre-Invocation  
02320 (0X0910) MTStartReadRequest Pre-Invocation  
02321 (0X0911) MTEndReadRequest Pre-Invocation  
02322 (0X0912) MTStartWriteRequest Pre-Invocation  
02323 (0X0913) MTEndWriteRequest Pre-Invocation  
02324 (0X0914) MTLongByteSwap Pre-Invocation

-----

## Trace Events for PMGPI Major Code: 0X00C5, Sorted by Tracepoint

GPIASSOCIATE 00103 (0X0067)  
GPIBEGINAREA 01106 (0X0452)  
GPIBEGINELEMENT 00500 (0X01F4)  
GPIBEGINPATH 00800 (0X0320)  
GPIBITBLT 01802 (0X070A)  
GPIBOX 01212 (0X04BC)  
GPICALLSEGMENTMATRIX 00605 (0X025D)  
GPICCHARSTRING 01416 (0X0588)  
GPICCHARSTRINGAT 01417 (0X0589)  
GPICCHARSTRINGPOS 01414 (0X0586)  
GPICCHARSTRINGPOSAT 01415 (0X0587)  
GPICLOSEFIGURE 00802 (0X0322)  
GPICLOSESEGMENT 00301 (0X012D)  
GPICOMBINEREGION 01903 (0X076F)  
GPICOMMENT 01604 (0X0644)  
GPICONVERT 00602 (0X025A)  
GPICOPYMETAFILE 02104 (0X0838)  
GPICORRELATECHAIN 00201 (0X00C9)  
GPICORRELATEFROM 00210 (0X00D2)  
GPICORRELATESEGMENT 00211 (0X00D3)  
GPICREATEBITMAP 01804 (0X070C)  
GPICREATELOGCOLORTABLE 00900 (0X0384)  
GPICREATELOGFONT 01705 (0X06A9)  
GPICREATEPS 00101 (0X0065)  
GPICREATEREGION 01900 (0X076C)  
GPIDELETEBITMAP 01800 (0X0708)  
GPIDELETEELEMENT 00505 (0X01F9)  
GPIDELETEELEMENTRANGE 00506 (0X01FA)  
GPIDELETEELEMENTSBETWEENLABELS 00507 (0X01FB)  
GPIDELETEMETAFILE 02103 (0X0837)  
GPIDELETESEGMENT 00302 (0X012E)  
GPIDELETESEGMENTS 00309 (0X0135)  
GPIDELETESETID 01700 (0X06A4)  
GPIDESTROYPS 00102 (0X0066)  
GPIDESTROYREGION 01902 (0X076E)  
GPIDRAWCHAIN 00402 (0X0192)  
GPIDRAWDYNAMICS 00405 (0X0195)  
GPIDRAWFROM 00403 (0X0193)  
GPIDRAWSEGMENT 00404 (0X0194)  
GPIELEMENT 00503 (0X01F7)  
GPIENDAREA 01107 (0X0453)  
GPIENDELEMENT 00501 (0X01F5)  
GPIENDPATH 00801 (0X0321)  
GPIEQUALREGION 01904 (0X0770)  
GPIERASE 00106 (0X006A)  
GPERRORSEGMENTDATA 00111 (0X006F)  
GPIEXCLUDECLIPRECTANGLE 02004 (0X07D4)  
GPIFILLPATH 00804 (0X0324)  
GPIFULLARC 01303 (0X0517)  
GPIGETDATA 00400 (0X0190)  
GPIIMAGE 01600 (0X0640)

GPIINTERSECTCLIPRECTANGLE 02003 (0X07D3)  
GPILABEL 00502 (0X01F6)  
GPILINE 01214 (0X04BE)  
GPILOADFONTS 01703 (0X06A7)  
GPILOADMETAFILE 02100 (0X0834)  
GPIMARKER 01506 (0X05E2)  
GPIMODIFYPATH 00803 (0X0323)  
GPIMOVE 01213 (0X04BD)  
GPIOFFSETCLIPREGION 02005 (0X07D5)  
GPIOFFSETELEMENTPOINTER 00512 (0X0200)  
GPIOFFSETREGION 01905 (0X0771)  
GPIOPENSEGMENT 00300 (0X012C)  
GPIPAINTREGION 01910 (0X0776)  
GPIPARTIALARC 01304 (0X0518)  
GPIPLAYMETAFILE 02101 (0X0835)  
GPIPOINTARC 01302 (0X0516)  
GPIPOLYFILLET 01307 (0X051B)  
GPIPOLYFILLETSARP 01305 (0X0519)  
GPIPOLYLINE 01215 (0X04BF)  
GPIPOLYMARKER 01507 (0X05E3)  
GPIPOLYSPLINE 01306 (0X051A)  
GPIPOP 01601 (0X0641)  
GPIPTINREGION 01906 (0X0772)  
GPIPTVISIBLE 01602 (0X0642)  
GPIPUTDATA 00401 (0X0191)  
GPIQUERYARCPARAMS 01301 (0X0515)  
GPIQUERYATTRMODE 01011 (0X03F3)  
GPIQUERYATTRS 01009 (0X03F1)  
GPIQUERYBACKCOLOR 01003 (0X03EB)  
GPIQUERYBACKMIX 01007 (0X03EF)  
GPIQUERYBITMAPBITS 01809 (0X0711)  
GPIQUERYBITMAPDIMENSION 01806 (0X070E)  
GPIQUERYBITMAPHANDLE 01814 (0X0716)  
GPIQUERYBITMAPPARAMETERS 01808 (0X0710)  
GPIQUERYBOUNDARYDATA 00208 (0X00D0)  
GPIQUERYCHARANGLE 01407 (0X057F)  
GPIQUERYCHARBOX 01405 (0X057D)  
GPIQUERYCHARDIRECTION 01411 (0X0583)  
GPIQUERYCHARMODE 01413 (0X0585)  
GPIQUERYCHARSET 01403 (0X057B)  
GPIQUERYCHARSHEAR 01409 (0X0581)  
GPIQUERYCHARSTRINGPOS 01418 (0X058A)  
GPIQUERYCHARSTRINGPOSAT 01419 (0X058B)  
GPIQUERYCLIPBOX 02002 (0X07D2)  
GPIQUERYCLIPREGION 02001 (0X07D1)  
GPIQUERYCOLOR 01001 (0X03E9)  
GPIQUERYCOLORDATA 00903 (0X0387)  
GPIQUERYCOLORINDEX 00907 (0X038B)  
GPIQUERYCP 01711 (0X06AF)  
GPIQUERYCURRENTPOSITION 01211 (0X04BB)  
GPIQUERYDEFAULTVIEWMATRIX 00606 (0X025E)  
GPIQUERYDEFCHARBOX 01401 (0X0579)  
GPIQUERYDEVICE 00107 (0X006B)  
GPIQUERYDEVICEBITMAPFORMATS 01807 (0X070F)  
GPIQUERYDRAWCONTROL 00112 (0X0070)  
GPIQUERYDRAWINGMODE 00114 (0X0072)  
GPIQUERYEDITMODE 00508 (0X01FC)  
GPIQUERYELEMENT 00504 (0X01F8)  
GPIQUERYELEMENTPOINTER 00510 (0X01FE)  
GPIQUERYELEMENTTYPE 00513 (0X0201)  
GPIQUERYFONTFILEDESCRIPTORS 01712 (0X06B0)  
GPIQUERYFONTMETRICS 01707 (0X06AB)  
GPIQUERYFONTS 01706 (0X06AA)  
GPIQUERYGRAPHICSFIELD 00701 (0X02BD)  
GPIQUERYINITIALSEGMENTATTRS 00303 (0X012F)  
GPIQUERYKERNINGPAIRS 01708 (0X06AC)  
GPIQUERYLINEEND 01207 (0X04B7)  
GPIQUERYLINEJOIN 01209 (0X04B9)  
GPIQUERYLINETYPE 01201 (0X04B1)  
GPIQUERYLINEWIDTH 01203 (0X04B3)  
GPIQUERYLINEWIDTHGEOM 01205 (0X04B5)  
GPIQUERYLOGCOLORTABLE 00904 (0X0388)  
GPIQUERYMARKER 01503 (0X05DF)



GPIQUERYMARKERBOX 01505 (0X05E1)  
GPIQUERYMARKERSET 01501 (0X05DD)  
GPIQUERYMETAFILEBITS 02106 (0X083A)  
GPIQUERYMETAFILELENGTH 02105 (0X0839)  
GPIQUERYMIX 01005 (0X03ED)  
GPIQUERYMODELTRANSFORMMATRIX 00603 (0X025B)  
GPIQUERYNEARESTCOLOR 00906 (0X038A)  
GPIQUERYNUMBERSETIDS 01701 (0X06A5)  
GPIQUERYPAGEVIEWPORT 00608 (0X0260)  
GPIQUERYPATTERN 01105 (0X0451)  
GPIQUERYPATTERNREFPOINT 01103 (0X044F)  
GPIQUERYPATTERNSET 01101 (0X044D)  
GPIQUERYPEL 01812 (0X0714)  
GPIQUERYPICKAPERTUREPOSITION 00206 (0X00CE)  
GPIQUERYPICKAPERTURESIZE 00204 (0X00CC)  
GPIQUERYPSP 00110 (0X006E)  
GPIQUERYREALCOLORS 00905 (0X0389)  
GPIQUERYREGIONBOX 01908 (0X0774)  
GPIQUERYREGIONRECTS 01909 (0X0775)  
GPIQUERYRGBCOLOR 00908 (0X038C)  
GPIQUERYSEGMENTATTRS 00305 (0X0131)  
GPIQUERYSEGMENTNAMES 00310 (0X0136)  
GPIQUERYSEGMENTPRIORITY 00307 (0X0133)  
GPIQUERYSEGMENTTRANSFORMMATRIX 00600 (0X0258)  
GPIQUERYSETIDS 01702 (0X06A6)  
GPIQUERYSTOPDRAW 00116 (0X0074)  
GPIQUERYTAG 00202 (0X00CA)  
GPIQUERYTEXTBOX 01400 (0X0578)  
GPIQUERYVIEWINGLIMITS 00703 (0X02BF)  
GPIQUERYVIEWINGTRANSFORMMATRIX 00610 (0X0262)  
GPIQUERYWIDTHTABLE 01709 (0X06AD)  
GPIREALIZECOLORTABLE 00901 (0X0385)  
GPIRECTINREGION 01907 (0X0773)  
GPIRECTVISIBLE 01603 (0X0643)  
GPIREMOVEDYNAMICS 00406 (0X0196)  
GPIRESETBOUNDARYDATA 00209 (0X00D1)  
GPIRESETPS 00108 (0X006C)  
GPIRESTOREPS 00104 (0X0068)  
GPISAVEMETAFILE 02102 (0X0836)  
GPISAVEPS 00105 (0X0069)  
GPISETARCPARAMS 01300 (0X0514)  
GPISETATTRMODE 01010 (0X03F2)  
GPISETATTRS 01008 (0X03F0)  
GPISETBACKCOLOR 01002 (0X03EA)  
GPISETBACKMIX 01006 (0X03EE)  
GPISETBITMAP 01801 (0X0709)  
GPISETBITMAPBITS 01810 (0X0712)  
GPISETBITMAPDIMENSION 01805 (0X070D)  
GPISETBITMAPID 01813 (0X0715)  
GPISETCHARANGLE 01406 (0X057E)  
GPISETCHARBOX 01404 (0X057C)  
GPISETCHARDIRECTION 01410 (0X0582)  
GPISETCHARMODE 01412 (0X0584)  
GPISETCHARSET 01402 (0X057A)  
GPISETCHARSHEAR 01408 (0X0580)  
GPISETCLIPPATH 00805 (0X0325)  
GPISETCLIPREGION 02000 (0X07D0)  
GPISETCOLOR 01000 (0X03E8)  
GPISETCP 01710 (0X06AE)  
GPISETCURRENTPOSITION 01210 (0X04BA)  
GPISETDEFAULTVIEWMATRIX 00607 (0X025F)  
GPISETDRAWCONTROL 00113 (0X0071)  
GPISETDRAWINGMODE 00115 (0X0073)  
GPISETEDITMODE 00509 (0X01FD)  
GPISETELEMENTPOINTER 00511 (0X01FF)  
GPISETELEMENTPOINTERATLABEL 00514 (0X0202)  
GPISETGRAPHICSFIELD 00700 (0X02BC)  
GPISETINITIALSEGMENTATTRS 00304 (0X0130)  
GPISETLINEEND 01206 (0X04B6)  
GPISETLINEJOIN 01208 (0X04B8)  
GPISETLINETYPE 01200 (0X04B0)  
GPISETLINEWIDTH 01202 (0X04B2)  
GPISETLINEWIDTHGEOM 01204 (0X04B4)

GPISETMARKER 01502 (0X05DE)  
GPISETMARKERBOX 01504 (0X05E0)  
GPISETMARKERSET 01500 (0X05DC)  
GPISETMETAFILEBITS 02107 (0X083B)  
GPISETMIX 01004 (0X03EC)  
GPISETMODELTRANSFORMMATRIX 00604 (0X025C)  
GPISETPAGEVIEWPORT 00609 (0X0261)  
GPISETPATTERN 01104 (0X0450)  
GPISETPATTERNREFPOINT 01102 (0X044E)  
GPISETPATTERNSET 01100 (0X044C)  
GPISETPEL 01811 (0X0713)  
GPISETPICKAPERTUREPOSITION 00207 (0X00CF)  
GPISETPICKAPERTURESIZE 00205 (0X00CD)  
GPISETPS 00109 (0X006D)  
GPISETREGION 01901 (0X076D)  
GPISETSEGMENTATTRS 00306 (0X0132)  
GPISETSEGMENTPRIORITY 00308 (0X0134)  
GPISETSEGMENTTRANSFORMMATRIX 00601 (0X0259)  
GPISETSTOPDRAW 00117 (0X0075)  
GPISETTAG 00203 (0X00CB)  
GPISETVIEWINGLIMITS 00702 (0X02BE)  
GPISETVIEWINGTRANSFORMMATRIX 00611 (0X0263)  
GPISTROKEPATH 00806 (0X0326)  
GPIUNLOADFONTS 01704 (0X06A8)  
GPIUNREALIZECOLORTABLE 00902 (0X0386)  
GPIWCBITBLT 01803 (0X070B)  
MTASSOCIATEMETA 02309 (0X0905)  
MTBITBLTMETA 02311 (0X0907)  
MTCLIPREGIONMETA 02315 (0X090B)  
MTCREATELOGCOLORTABLEMETA 02303 (0X08FF)  
MTDELETESETIDMETA 02305 (0X0901)  
MTDEVESCAPE 02317 (0X090D)  
MTDISPLAYFLAGMETA 02302 (0X08FE)  
MTENABLEKERNINGMETA 02301 (0X08FD)  
MTENDREADREQUEST 02321 (0X0911)  
MTENDWRITEREQUEST 02323 (0X0913)  
MTERASEMETA 02308 (0X0904)  
MTIDPSTOREMETA 02300 (0X08FC)  
MTLONGBYTESWAP 02324 (0X0914)  
MTPAINTREGIONMETA 02316 (0X090C)  
MRESETMETA 02307 (0X0903)  
MTRESTOREPS 02318 (0X090E)  
MTSAVEPS 02319 (0X090F)  
MTSELECTCLIPMETA 02314 (0X090A)  
MTSETCODEPAGEMETA 02304 (0X0900)  
MTSETGRAPHICSFIELDMETA 02306 (0X0902)  
MTSETPELMETA 02313 (0X0909)  
MTSTARTREADREQUEST 02320 (0X0910)  
MTSTARTWRITEREQUEST 02322 (0X0912)  
MTVERIFYPAGEUNITS 02310 (0X0906)  
MTWCBITBLTMETA 02312 (0X0908)

-----

## PMGPI Major Code: 0X00C5 Minor Code: 101 (0X0065)

### Description

GpiCreatePS Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPICREATEPS

### Minor Code

101 (0X0065)

### Trace Groups

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Options = %d, WidthHeight = %a, HDC = %d, Hab = %d  
WidthHeight->cx = %d, WidthHeight->cy = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 102 (0X0066)

**Description**

GpiDestroyPS Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDESTROYPS

**Minor Code**

102 (0X0066)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 103 (0X0067)

**Description**

GpiAssociate Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIASSOCIATE

**Minor Code**

103 (0X0067)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%DcH = %d, GpiH = %d

-----

# PMGPI Major Code: 0X00C5 Minor Code: 104 (0X0068)

<u>Description</u>	GpiRestorePS Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIRESTOREPS
<u>Minor Code</u>	104 (0X0068)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Psid = %d, GpiH = %d

# PMGPI Major Code: 0X00C5 Minor Code: 105 (0X0069)

<u>Description</u>	GpiSavePS Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPISAVEPS
<u>Minor Code</u>	105 (0X0069)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%GpiH = %d

# PMGPI Major Code: 0X00C5 Minor Code: 106 (0X006A)

<u>Description</u>	GpiErase Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIERASE
<u>Minor Code</u>	106 (0X006A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 107 (0X006B)

**Description**

GpiQueryDevice Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEVICE

**Minor Code**

107 (0X006B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 108 (0X006C)

**Description**

GpiResetPS Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIRESETPS

**Minor Code**

108 (0X006C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Psid = %d, GpiH = %d

---

# PMGPI Major Code: 0X00C5 Minor Code: 109 (0X006D)

<u>Description</u>	GpiSetPS Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPISETPS
<u>Minor Code</u>	109 (0X006D)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<div>%Options = %d, PresentationSize = %a, GpiH = %d PresentationSize-&gt;cx = %d, PresentationSize-&gt;cy = %d</div>

# PMGPI Major Code: 0X00C5 Minor Code: 110 (0X006E)

<u>Description</u>	GpiQueryPS Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPS
<u>Minor Code</u>	110 (0X006E)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<div>%GpiH = %d</div>

# PMGPI Major Code: 0X00C5 Minor Code: 111 (0X006F)

<u>Description</u>	GpiErrorSegmentData Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIERRORSEGMENTDATA

<u>Minor Code</u>	111 (0X006F)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 112 (0X0070)

<u>Description</u>	GpiQueryDrawControl Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDRAWCONTROL
<u>Minor Code</u>	112 (0X0070)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Control = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 113 (0X0071)

<u>Description</u>	GpiSetDrawControl Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPISETDRAWCONTROL
<u>Minor Code</u>	113 (0X0071)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Value = %d, Control = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 114 (0X0072)

<u>Description</u>	GpiQueryDrawingMode Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDRAWINGMODE
<u>Minor Code</u>	114 (0X0072)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 115 (0X0073)

<u>Description</u>	GpiSetDrawingMode Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPISETDRAWINGMODE
<u>Minor Code</u>	115 (0X0073)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Mode = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 116 (0X0074)

<u>Description</u>	GpiQueryStopDraw Pre-Invocation
<u>Tracepoint</u>	



Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSTOPDRAW

**Minor Code**

116 (0X0074)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 117 (0X0075)

**Description**

GpiSetStopDraw Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETSTOPDRAW

**Minor Code**

117 (0X0075)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Value = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 201 (0X00C9)

**Description**

GpiCorrelateChain Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPICORRELATECHAIN

**Minor Code**

201 (0X00C9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Depth = %d, MaxHits = %d, PickWindowPos = %a

Ctype = %d, GpiH = %d

PickWindowPos->x = %d, PickWindowPos->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 202 (0X00CA)

### Description

GpiQueryTag Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYTAG

### Minor Code

202 (0X00CA)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 203 (0X00CB)

### Description

GpiSetTag Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETTAG

### Minor Code

203 (0X00CB)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%V = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 204 (0X00CC)

<b><u>Description</u></b>	GpiQueryPickApertureSize Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPICKAPERTURESIZE
<b><u>Minor Code</u></b>	204 (0X00CC)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 205 (0X00CD)

<b><u>Description</u></b>	GpiSetPickApertureSize Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETPICKAPERTURESIZE
<b><u>Minor Code</u></b>	205 (0X00CD)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%ApertureSize = %a, Options = %d, GpiH = %d ApertureSize.cx = %d, ApertureSize.cy = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 206 (0X00CE)

<b><u>Description</u></b>	GpiQueryPickAperturePosition Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPICKAPERTUREPOSITION
<b><u>Minor Code</u></b>	206 (0X00CE)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 207 (0X00CF)

**Description**

GpiSetPickAperturePosition Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETPICKAPERTUREPOSITION

**Minor Code**

207 (0X00CF)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%AperturePosition = %a, GpiH = %d

AperturePosition->x = %d, AperturePosition->y = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 208 (0X00D0)

**Description**

GpiQueryBoundaryData Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBOUNDARYDATA

**Minor Code**

208 (0X00D0)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 209 (0X00D1)

<b><u>Description</u></b>	GpiResetBoundaryData Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIRESETBOUNDARYDATA
<b><u>Minor Code</u></b>	209 (0X00D1)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 210 (0X00D2)

<b><u>Description</u></b>	GpiCorrelateFrom Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPICORRELATEFROM
<b><u>Minor Code</u></b>	210 (0X00D2)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%MaxDepth = %d, MaxHits = %d, PickPosition = %a Type = %d, LastSegment = %d, FirstSegment = %d, GpiH = %d Pick->x = %d, Pick->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 211 (0X00D3)

<b><u>Description</u></b>	GpiCorrelateSegment Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPICORRELATESEGMENT
<b><u>Minor Code</u></b>	

211 (0X00D3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MaxDepth = %d, MaxHits = %d, PickPosition = %a

Type = %d, Segment = %d, GpiH = %d

Pick->x = %d, Pick->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 300 (0X012C)

**Description**

GpiOpenSegment Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIOPENSEGMENT

**Minor Code**

300 (0X012C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%SegmentID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 301 (0X012D)

**Description**

GpiCloseSegment Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPICLOSESEGMENT

**Minor Code**

301 (0X012D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 302 (0X012E)

### Description

GpiDeleteSegment Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETESEGMENT

### Minor Code

302 (0X012E)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%SegmentID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 303 (0X012F)

### Description

GpiQueryInitialSegmentAttrs Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYINITIALSEGMENTATTRS

### Minor Code

303 (0X012F)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Attribute = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 304 (0X0130)

### Description

GpiSetInitialSegmentAttrs Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETINITIALSEGMENTATTRS
<b><u>Minor Code</u></b>	304 (0X0130)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Value = %d, Attribute = %d, GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 305 (0X0131)

<b><u>Description</u></b>	GpiQuerySegmentAttrs Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTATTRS
<b><u>Minor Code</u></b>	305 (0X0131)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Attribute = %d, SegmentID = %d, GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 306 (0X0132)

<b><u>Description</u></b>	GpiSetSegmentAttrs Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETSEGMENTATTRS
<b><u>Minor Code</u></b>	306 (0X0132)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	



%Value = %d, Attribute = %d, SegmentID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 307 (0X0133)

### Description

GpiQuerySegmentPriority Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTPRIORITY

### Minor Code

307 (0X0133)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Order = %d, ReferenceSegmentID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 308 (0X0134)

### Description

GpiSetSegmentPriority Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETSEGMENTPRIORITY

### Minor Code

308 (0X0134)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Order = %d, ReferenceSegmentID = %d, SegmentID = %d,

GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 309 (0X0135)

**Description**

GpiDeleteSegments Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETESEGMENTS

**Minor Code**

309 (0X0135)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LastSegment = %d, FirstSegment = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 310 (0X0136)

**Description**

GpiQuerySegmentNames Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTNAMES

**Minor Code**

310 (0X0136)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MaxNumber = %d, LastSegment = %d,

FirstSegment = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 400 (0X0190)

**Description**

GpiGetData Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIGETDATA

**Minor Code**

400 (0X0190)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ReadLen = %d, Control = %d, Offset = %a,  
SegmentName = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 401 (0X0191)

**Description**

GpiPutData Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIPUTDATA

**Minor Code**

401 (0X0191)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%DataPtr = %a, LenPtr = %d, Control = %d, GpiH = %d  
%DataPtr = %b%b %b%b %b%b %b%b %b%b

-----

PMGPI Major Code: 0X00C5 Minor Code: 402 (0X0192)

**Description**

GpiDrawChain Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWCHAIN

**Minor Code**

402 (0X0192)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

# PMGPI Major Code: 0X00C5 Minor Code: 403 (0X0193)

<u>Description</u>	GpiDrawFrom Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWFROM
<u>Minor Code</u>	403 (0X0193)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%LastSegment = %d, FirstDegment = %d, GpiH = %d

# PMGPI Major Code: 0X00C5 Minor Code: 404 (0X0194)

<u>Description</u>	GpiDrawSegment Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWSEGMENT
<u>Minor Code</u>	404 (0X0194)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%SegmentID = %d, GpiH = %d

# PMGPI Major Code: 0X00C5 Minor Code: 405 (0X0195)

<u>Description</u>	GpiDrawDynamics Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIDRAWDYNAMICS
<u>Minor Code</u>	

405 (0X0195)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 406 (0X0196)

**Description**

GpiRemoveDynamics Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIREMOVEDYNAMICS

**Minor Code**

406 (0X0196)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LastSegmentID = %d, FirstSegmentID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 500 (0X01F4)

**Description**

GpiBeginElement Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIBEGINELEMENT

**Minor Code**

500 (0X01F4)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%desc = %a, type = %d, GpiH = %d

desc-> = %s

-----

PMGPI Major Code: 0X00C5 Minor Code: 501 (0X01F5)

<u>Description</u>	GpiEndElement Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIENDELEMENT
<u>Minor Code</u>	501 (0X01F5)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 502 (0X01F6)

<u>Description</u>	GpiLabel Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPILABEL
<u>Minor Code</u>	502 (0X01F6)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Label = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 503 (0X01F7)

<u>Description</u>	GpiElement Pre-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: PMGPI.GPIELEMENT

**Minor Code**

503 (0X01F7)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Data = %a, Length = %d, Desc = %a, Type = %d, GpiH = %d

Data = %s

Desc = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 504 (0X01F8)

**Description**

GpiQueryElement Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYELEMENT

**Minor Code**

504 (0X01F8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MaxLength = %d, Offset = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 505 (0X01F9)

**Description**

GpiDeleteElement Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEELEMENT

**Minor Code**

505 (0X01F9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 506 (0X01FA)

**Description**

GpiDeleteElementRange Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEELEMENTRANGE

**Minor Code**

506 (0X01FA)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LastElement = %d, FirstElement = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 507 (0X01FB)

**Description**

GpiDeleteElementsBetweenLabels Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEELEMENTSBETWEENLABELS

**Minor Code**

507 (0X01FB)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LastLabel = %d, FirstLabel = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 508 (0X01FC)



<b><u>Description</u></b>	GpiQueryEditMode Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYEDITMODE
<b><u>Minor Code</u></b>	508 (0X01FC)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 509 (0X01FD)

<b><u>Description</u></b>	GpiSetEditMode Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETEDITMODE
<b><u>Minor Code</u></b>	509 (0X01FD)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%EditMode = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 510 (0X01FE)

<b><u>Description</u></b>	GpiQueryElementPointer Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYELEMENTPOINTER
<b><u>Minor Code</u></b>	510 (0X01FE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 511 (0X01FF)

**Description**

GpiSetElementPointer Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETELEMENTPOINTER

**Minor Code**

511 (0X01FF)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ElementNumber = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 512 (0X0200)

**Description**

GpiOffsetElementPointer Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIOFFSETELEMENTPOINTER

**Minor Code**

512 (0X0200)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Offset = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 513 (0X0201)

<b><u>Description</u></b>	GpiQueryElementType Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYELEMENTTYPE
<b><u>Minor Code</u></b>	513 (0X0201)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	%Data Length = %d
	GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 514 (0X0202)

<b><u>Description</u></b>	GpiSetElementPointerAtLabel Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETELEMENTPOINTERATLABEL
<b><u>Minor Code</u></b>	514 (0X0202)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	
	%Label = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 600 (0X0258)

<b><u>Description</u></b>	GpiQuerySegmentTransformMatrix Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSEGMENTTRANSFORMMATRIX
<b><u>Minor Code</u></b>	600 (0X0258)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Count = %d, SegmentID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 601 (0X0259)

**Description**  
GpiSetSegmentTransformMatrix Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPISETSEGMENTTRANSFORMMATRIX

**Minor Code**  
601 (0X0259)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Options = %d, Matrix = %a, Count = %d,  
SegmentID = %d, GpiH = %d  
  
Matrix->fxM11 = %d, Matrix->fxM12 = %d  
Matrix->IM13 = %d, Matrix->fxM21 = %d  
Matrix->fxM22 = %d, Matrix->IM23 = %d  
Matrix->IM31 = %d, Matrix->IM32 = %d  
Matrix->IM33 = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 602 (0X025A)

**Description**  
GpiConvert Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPICONVERT

**Minor Code**  
602 (0X025A)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**

%Count = %d, TargetCoordSpace = %d,  
SrcCoordSpace = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 603 (0X025B)

**Description**

GpiQueryModelTransformMatrix Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMODELTRANSFORMMATRIX

**Minor Code**

603 (0X025B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Count = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 604 (0X025C)

**Description**

GpiSetModelTransformMatrix Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETMODELTRANSFORMMATRIX

**Minor Code**

604 (0X025C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Options = %d, Matrix = %a, Count = %d, GpiH = %d  
  
Matrix->fxM11 = %d, Matrix->fxM12 = %d  
  
Matrix->IM13 = %d, Matrix->fxM21 = %d  
  
Matrix->fxM22 = %d, Matrix->IM23 = %d

Matrix->IM31 = %d, Matrix->IM32 = %d  
Matrix->IM33 = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 605 (0X025D)

### Description

GpiCallSegmentMatrix Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPICALLSEGMENTMATRIX

### Minor Code

605 (0X025D)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Options = %d, Matrix = %a, Count = %d,

SegmentID = %d, GpiH = %d

Matrix->fxM11 = %d, Matrix->fxM12 = %d

Matrix->IM13 = %d, Matrix->fxM21 = %d

Matrix->fxM22 = %d, Matrix->IM23 = %d

Matrix->IM31 = %d, Matrix->IM32 = %d

Matrix->IM33 = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 606 (0X025E)

### Description

GpiQueryDefaultViewMatrix Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEFAULTVIEWMATRIX

### Minor Code

606 (0X025E)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Count = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 607 (0X025F)

### Description

GpiSetDefaultViewMatrix Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETDEFAULTVIEWMATRIX

### Minor Code

607 (0X025F)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Options = %d, Matrix = %a, Count = %d, GpiH = %d

Matrix->fxM11 = %d, Matrix->fxM12 = %d

Matrix->IM13 = %d, Matrix->fxM21 = %d

Matrix->fxM22 = %d, Matrix->IM23 = %d

Matrix->IM31 = %d, Matrix->IM32 = %d

Matrix->IM33 = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 608 (0X0260)

### Description

GpiQueryPageViewport Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPAGEVIEWPORT

### Minor Code

608 (0X0260)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%GpiH = %d

-----

# PMGPI Major Code: 0X00C5 Minor Code: 609 (0X0261)

<u>Description</u>	GpiSetPageViewport Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPISETPAGEVIEWPORT
<u>Minor Code</u>	609 (0X0261)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<div>%Viewport = %a, GpiH = %d Viewport-&gt;xLeft = %d, Viewport-&gt;yBottom = %d Viewport-&gt;xRight = %d, Viewport-&gt;yTop = %d</div>

# PMGPI Major Code: 0X00C5 Minor Code: 610 (0X0262)

<u>Description</u>	GpiQueryViewingTransformMatrix Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYVIEWINGTRANSFORMMATRIX
<u>Minor Code</u>	610 (0X0262)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<div>%Count = %d, GpiH = %d</div>

# PMGPI Major Code: 0X00C5 Minor Code: 611 (0X0263)

<u>Description</u>	GpiSetViewingTransformMatrix Pre-Invocation
--------------------	---



**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETVIEWINGTRANSFORMMATRIX

**Minor Code**

611 (0X0263)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Options = %d, Matrix = %a, Count = %d, GpiH = %d

Matrix->fxM11 = %d, Matrix->fxM12 = %d

Matrix->IM13 = %d, Matrix->fxM21 = %d

Matrix->fxM22 = %d, Matrix->IM23 = %d

Matrix->IM31 = %d, Matrix->IM32 = %d

Matrix->IM33 = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 700 (0X02BC)

**Description**

GpiSetGraphicsField Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETGRAPHICSFIELD

**Minor Code**

700 (0X02BC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Field = %a, GpiH = %d

Field->xLeft = %d, Field->yBottom = %d

Field->xRight = %d, Field->yTop = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 701 (0X02BD)

**Description**

GpiQueryGraphicsField Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYGRAPHICSFIELD
<b><u>Minor Code</u></b>	701 (0X02BD)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 702 (0X02BE)

<b><u>Description</u></b>	GpiSetViewingLimits Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETVIEWINGLIMITS
<b><u>Minor Code</u></b>	702 (0X02BE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Limits = %d, GpiH = %d Limits->xLeft = %d, Limits->yBottom = %d Limits->xRight = %d, Limits->yTop = %d

## PMGPI Major Code: 0X00C5 Minor Code: 703 (0X02BF)

<b><u>Description</u></b>	GpiQueryViewingLimits Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYVIEWINGLIMITS
<b><u>Minor Code</u></b>	703 (0X02BF)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 800 (0X0320)

**Description**

GpiBeginPath Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIBEGINPATH

**Minor Code**

800 (0X0320)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%PathID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 801 (0X0321)

**Description**

GpiEndPath Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIENDPATH

**Minor Code**

801 (0X0321)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 802 (0X0322)

<b><u>Description</u></b>	GpiCloseFigure Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPICLOSEFIGURE
<b><u>Minor Code</u></b>	802 (0X0322)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 803 (0X0323)

<b><u>Description</u></b>	GpiModifyPath Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIMODIFYPATH
<b><u>Minor Code</u></b>	803 (0X0323)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Mode = %d, PathID = %d, GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 804 (0X0324)

<b><u>Description</u></b>	GpiFillPath Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIFILLPATH
<b><u>Minor Code</u></b>	804 (0X0324)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Options = %d, PathID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 805 (0X0325)

**Description**

GpiSetClipPath Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETCLIPPATH

**Minor Code**

805 (0X0325)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Options = %d, PathID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 806 (0X0326)

**Description**

GpiStrokePath Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISTROKEPATH

**Minor Code**

806 (0X0326)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Options = %d, PathID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 900 (0X0384)

<b><u>Description</u></b>	GpiCreateLogColorTable Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPICREATELOGCOLORTABLE
<b><u>Minor Code</u></b>	900 (0X0384)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Table=%a, Count=%d, Start=%d, Format=%d, Options=%d, GpiH = %d Table->field1 = %d, Table->field2 = %d Table->field3 = %d, Table->field4 = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 901 (0X0385)

<b><u>Description</u></b>	GpiRealizeColorTable Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIREALIZECOLORTABLE
<b><u>Minor Code</u></b>	901 (0X0385)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 902 (0X0386)

<b><u>Description</u></b>	GpiUnrealizeColorTable Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIUNREALIZECOLORTABLE

<u>Minor Code</u>	902 (0X0386)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 903 (0X0387)

<u>Description</u>	GpiQueryColorData Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCOLORDATA
<u>Minor Code</u>	903 (0X0387)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Count = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 904 (0X0388)

<u>Description</u>	GpiQueryLogColorTable Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLOGCOLORTABLE
<u>Minor Code</u>	904 (0X0388)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Count = %d, Start = %d, Options = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 905 (0X0389)

<u>Description</u>	GpiQueryRealColors Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYREALCOLORS
<u>Minor Code</u>	905 (0X0389)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Count = %d, Start = %d, Options = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 906 (0X038A)

<u>Description</u>	GpiQueryNearestColor Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYNEARESTCOLOR
<u>Minor Code</u>	906 (0X038A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%RequiredColor = %d, Options = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 907 (0X038B)

<u>Description</u>	GpiQueryColorIndex Pre-Invocation
<u>Tracepoint</u>	



Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCOLORINDEX

**Minor Code**

907 (0X038B)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%RGBColor = %d, Options = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 908 (0X038C)

**Description**

GpiQueryRGBColor Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYRGBCOLOR

**Minor Code**

908 (0X038C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ColorIndex = %d, Options = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1000 (0X03E8)

**Description**

GpiSetColor Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETCOLOR

**Minor Code**

1000 (0X03E8)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Color = %d, GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 1001 (0X03E9)

**Description**

GpiQueryColor Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCOLOR

**Minor Code**

1001 (0X03E9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 1002 (0X03EA)

**Description**

GpiSetBackColor Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETBACKCOLOR

**Minor Code**

1002 (0X03EA)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Color = %d, GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 1003 (0X03EB)

**Description**

GpiQueryBackColor Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBACKCOLOR

**Minor Code**

1003 (0X03EB)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1004 (0X03EC)

**Description**

GpiSetMix Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETMIX

**Minor Code**

1004 (0X03EC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MixMode = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1005 (0X03ED)

**Description**

GpiQueryMix Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMIX

**Minor Code**

1005 (0X03ED)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1006 (0X03EE)

### Description

GpiSetBackMix Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETBACKMIX

### Minor Code

1006 (0X03EE)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%MixMode = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1007 (0X03EF)

### Description

GpiQueryBackMix Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBACKMIX

### Minor Code

1007 (0X03EF)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1008 (0X03F0)

### Description

GpiSetAttr Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETATTRS
<b><u>Minor Code</u></b>	1008 (0X03F0)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%DefMask = %d, AttrMask = %d, PrimType = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1009 (0X03F1)

<b><u>Description</u></b>	GpiQueryAttrs Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYATTRS
<b><u>Minor Code</u></b>	1009 (0X03F1)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%AttrMask = %d, PrimType = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1010 (0X03F2)

<b><u>Description</u></b>	GpiSetAttrMode Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETATTRMODE
<b><u>Minor Code</u></b>	1010 (0X03F2)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

%AttrMode = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1011 (0X03F3)

### Description

GpiQueryAttrMode Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYATTRMODE

### Minor Code

1011 (0X03F3)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1100 (0X044C)

### Description

GpiSetPatternSet Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETPATTERNSET

### Minor Code

1100 (0X044C)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%PatternSetID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1101 (0X044D)

### Description

GpiQueryPatternSet Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPATTERNSET

**Minor Code**

1101 (0X044D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1102 (0X044E)

**Description**

GpiSetPatternRefPoint Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETPATTERNREFPOINT

**Minor Code**

1102 (0X044E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%PatternRefPt = %a, GpiH = %d

PatternRefPt->x = %d, PatternRefPt->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1103 (0X044F)

**Description**

GpiQueryPatternRefPoint Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPATTERNREFPOINT

**Minor Code**

1103 (0X044F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1104 (0X0450)

**Description**

GpiSetPattern Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETPATTERN

**Minor Code**

1104 (0X0450)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%PatternSymbol = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1105 (0X0451)

**Description**

GpiQueryPattern Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPATTERN

**Minor Code**

1105 (0X0451)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1106 (0X0452)



<b><u>Description</u></b>	GpiBeginArea Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIBEGINAREA
<b><u>Minor Code</u></b>	1106 (0X0452)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%AreaOption = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1107 (0X0453)

<b><u>Description</u></b>	GpiEndArea Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIENDAREA
<b><u>Minor Code</u></b>	1107 (0X0453)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1200 (0X04B0)

<b><u>Description</u></b>	GpiSetLineType Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETLINETYPE
<b><u>Minor Code</u></b>	1200 (0X04B0)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LineType = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1201 (0X04B1)

**Description**

GpiQueryLineType Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINETYPE

**Minor Code**

1201 (0X04B1)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1202 (0X04B2)

**Description**

GpiSetLineWidth Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETLINEWIDTH

**Minor Code**

1202 (0X04B2)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LineWidth = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1203 (0X04B3)

<b><u>Description</u></b>	GpiQueryLineWidth Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEWIDTH
<b><u>Minor Code</u></b>	1203 (0X04B3)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1204 (0X04B4)

<b><u>Description</u></b>	GpiSetLineWidthGeom Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETLINEWIDTHGEOM
<b><u>Minor Code</u></b>	1204 (0X04B4)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%LineWidthGeom = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1205 (0X04B5)

<b><u>Description</u></b>	GpiQueryLineWidthGeom Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEWIDTHGEOM
<b><u>Minor Code</u></b>	1205 (0X04B5)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1206 (0X04B6)

**Description**

GpiSetLineEnd Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETLINEEND

**Minor Code**

1206 (0X04B6)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%LineEnd = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1207 (0X04B7)

**Description**

GpiQueryLineEnd Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEEND

**Minor Code**

1207 (0X04B7)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1208 (0X04B8)

<b><u>Description</u></b>	GpiSetLineJoin Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETLINEJOIN
<b><u>Minor Code</u></b>	1208 (0X04B8)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%LineJoin = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1209 (0X04B9)

<b><u>Description</u></b>	GpiQueryLineJoin Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYLINEJOIN
<b><u>Minor Code</u></b>	1209 (0X04B9)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1210 (0X04BA)

<b><u>Description</u></b>	GpiSetCurrentPosition Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETCURRENTPOSITION
<b><u>Minor Code</u></b>	1210 (0X04BA)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Point = %a, GpiH = %d

Point->x = %d, Point->y = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 1211 (0X04BB)

**Description**

GpiQueryCurrentPosition Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCURRENTPOSITION

**Minor Code**

1211 (0X04BB)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 1212 (0X04BC)

**Description**

GpiBox Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIBOX

**Minor Code**

1212 (0X04BC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%vRound = %d, hRound = %d, Point = %a,

Control = %d, GpiH = %d

Point->x = %d, Point->y = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1213 (0X04BD)

<u>Description</u>	GpiMove Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIMOVE
<u>Minor Code</u>	1213 (0X04BD)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %Point = %a, GpiH = %d Point->x = %d, Point->y = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1214 (0X04BE)

<u>Description</u>	GpiLine Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPILINE
<u>Minor Code</u>	1214 (0X04BE)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %Point = %a, GpiH = %d Point->x = %d, Point->y = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1215 (0X04BF)

<b><u>Description</u></b>	GpiPolyLine Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYLINE
<b><u>Minor Code</u></b>	1215 (0X04BF)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	<p>%Point = %a, Count = %d, GpiH = %d</p> <p>Point-&gt;x = %d, Point-&gt;y = %d</p>

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1300 (0X0514)

<b><u>Description</u></b>	GpiSetArcParams Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETARCPARAMS
<b><u>Minor Code</u></b>	1300 (0X0514)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	<p>%ArcParams = %a, GpiH = %d</p> <p>ArcParams-IP = %d, ArcParams-&gt;IQ = %d</p> <p>ArcParams-IR = %d, ArcParams-&gt;IS = %d</p>

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1301 (0X0515)

<b><u>Description</u></b>	GpiQueryArcParams Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYARCPARAMS
<b><u>Minor Code</u></b>	1301 (0X0515)



**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1302 (0X0516)

**Description**

GpiPointArc Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIPOINTARC

**Minor Code**

1302 (0X0516)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Point = %a, GpiH = %d

Point->x = %d, Point->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1303 (0X0517)

**Description**

GpiFullArc Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIFULLARC

**Minor Code**

1303 (0X0517)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Multiplier = %d, Control = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1304 (0X0518)

<u>Description</u>	GpiPartialArc Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIPARTIALARC
<u>Minor Code</u>	1304 (0X0518)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %SweepAngle=%d, StartAngle=%d, Multiplier=%d, Centre=%a, GpiH = %d Centre->x = %d, Centre->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1305 (0X0519)

<u>Description</u>	GpiPolyFilletSharp Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYFILLETSARP
<u>Minor Code</u>	1305 (0X0519)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %Sharpness = %a, Point = %a, Count = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1306 (0X051A)

<u>Description</u>	GpiPolySpline Pre-Invocation
--------------------	------------------------------

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYSPLINE
<b><u>Minor Code</u></b>	1306 (0X051A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Point = %a, Count = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1307 (0X051B)

<b><u>Description</u></b>	GpiPolyFillet Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYFILLET
<b><u>Minor Code</u></b>	1307 (0X051B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Point = %a, Count = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1400 (0X0578)

<b><u>Description</u></b>	GpiQueryTextBox Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYTEXTBOX
<b><u>Minor Code</u></b>	1400 (0X0578)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	

%Count2 = %d, String = %a, Count1 = %d, GpiH = %d  
String = %s

---

## PMGPI Major Code: 0X00C5 Minor Code: 1401 (0X0579)

### Description

GpiQueryDefCharBox Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEFCHARBOX

### Minor Code

1401 (0X0579)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 1402 (0X057A)

### Description

GpiSetCharSet Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARSET

### Minor Code

1402 (0X057A)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Lcid = %d, GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 1403 (0X057B)

<b><u>Description</u></b>	GpiQueryCharSet Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARSET
<b><u>Minor Code</u></b>	1403 (0X057B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1404 (0X057C)

<b><u>Description</u></b>	GpiSetCharBox Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARBOX
<b><u>Minor Code</u></b>	1404 (0X057C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Size = %a, GpiH = %d Size->cx = %d, Size->cy

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1405 (0X057D)

<b><u>Description</u></b>	GpiQueryCharBox Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARBOX
<b><u>Minor Code</u></b>	1405 (0X057D)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

PMGPI Major Code: 0X00C5 Minor Code: 1406 (0X057E)

**Description**

GpiSetCharAngle Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARANGLE

**Minor Code**

1406 (0X057E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Gradient = %a, GpiH = %d

Gradient->x = %d, Gradient->y = %d

---

PMGPI Major Code: 0X00C5 Minor Code: 1407 (0X057F)

**Description**

GpiQueryCharAngle Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARANGLE

**Minor Code**

1407 (0X057F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

---

PMGPI Major Code: 0X00C5 Minor Code: 1408 (0X0580)

<b><u>Description</u></b>	GpiSetCharShear Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARSHEAR
<b><u>Minor Code</u></b>	1408 (0X0580)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	<p>%Point = %a, GpiH = %d</p> <p>Point-&gt;x = %d, Point-&gt;y = %d</p>

## PMGPI Major Code: 0X00C5 Minor Code: 1409 (0X0581)

<b><u>Description</u></b>	GpiQueryCharShear Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARSHEAR
<b><u>Minor Code</u></b>	1409 (0X0581)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	<p>%GpiH = %d</p>

## PMGPI Major Code: 0X00C5 Minor Code: 1410 (0X0582)

<b><u>Description</u></b>	GpiSetCharDirection Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARDIRECTION
<b><u>Minor Code</u></b>	1410 (0X0582)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Direction = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1411 (0X0583)

**Description**

GpiQueryCharDirection Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARDIRECTION

**Minor Code**

1411 (0X0583)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1412 (0X0584)

**Description**

GpiSetCharMode Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETCHARMODE

**Minor Code**

1412 (0X0584)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Mode = %d, GpiH = %d

-----



# PMGPI Major Code: 0X00C5 Minor Code: 1413 (0X0585)

<u>Description</u>	GpiQueryCharMode Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARMODE
<u>Minor Code</u>	1413 (0X0585)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%GpiH = %d

# PMGPI Major Code: 0X00C5 Minor Code: 1414 (0X0586)

<u>Description</u>	GpiCharStringPos Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRINGPOS
<u>Minor Code</u>	1414 (0X0586)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Vector=%a, String=%a, Count=%d, Options=%d, Rect=%a, GpiH = %d String = %s Rect->xLeft = %d, Rect->yBottom = %d Rect->xRight = %d, Rect->yTop = %d

# PMGPI Major Code: 0X00C5 Minor Code: 1415 (0X0587)

**Description**

GpiCharStringPosAt Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRINGPOSAT

**Minor Code**

1415 (0X0587)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Vector=%a, String=%a, Count=%d, Options=%d,

Rect=%a, Point=%a, GpiH = %d

String = %s

Rect->xLeft = %d, Rect->yBottom = %d

Rect->xRight = %d, Rect->yTop = %d

Point->x = %d, Point->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1416 (0X0588)

**Description**

GpiCharString Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRING

**Minor Code**

1416 (0X0588)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%String = %a, Count = %d, GpiH = %d

String = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1417 (0X0589)

**Description**

GpiCharStringAt Pre-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPICHARSTRINGAT
<u>Minor Code</u>	1417 (0X0589)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%String = %a, Count = %d, Point = %a, GpiH = %d String = %s Point->x = %d, Point->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1418 (0X058A)

<u>Description</u>	GpiQueryCharStringPos Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARSTRINGPOS
<u>Minor Code</u>	1418 (0X058A)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%String = %a, Count = %d, Options = %d, GpiH = %d String = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1419 (0X058B)

<u>Description</u>	GpiQueryCharStringPosAt Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCHARSTRINGPOSAT
<u>Minor Code</u>	1419 (0X058B)
<u>Trace Groups</u>	

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%String = %a, Count = %d, Options = %d,

Start = %a, GpiH = %d

String = %s

Start->x = %d, Start->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1500 (0X05DC)

**Description**

GpiSetMarkerSet Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETMARKERSET

**Minor Code**

1500 (0X05DC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%MarkerSetID = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1501 (0X05DD)

**Description**

GpiQueryMarkerSet Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMARKERSET

**Minor Code**

1501 (0X05DD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1502 (0X05DE)

### Description

GpiSetMarker Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETMARKER

### Minor Code

1502 (0X05DE)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%MarkerSymbol = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1503 (0X05DF)

### Description

GpiQueryMarker Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMARKER

### Minor Code

1503 (0X05DF)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1504 (0X05E0)

### Description

GpiSetMarkerBox Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETMARKERBOX
<b><u>Minor Code</u></b>	1504 (0X05E0)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%MarkerBoxSize = %a, GpiH = %d MarkerBoxSize->cx = %d, MarkerBoxSize->cy = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1505 (0X05E1)

<b><u>Description</u></b>	GpiQueryMarkerBox Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMARKERBOX
<b><u>Minor Code</u></b>	1505 (0X05E1)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1506 (0X05E2)

<b><u>Description</u></b>	GpiMarker Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIMARKER
<b><u>Minor Code</u></b>	1506 (0X05E2)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

**Traced Parameters**

%Point = %a, GpiH = %d  
Point->x = %d, Point->y = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1507 (0X05E3)

**Description**

GpiPolyMarker Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIPOLYMARKER

**Minor Code**

1507 (0X05E3)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Point = %a, Count = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1600 (0X0640)

**Description**

GpiImage Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIIMAGE

**Minor Code**

1600 (0X0640)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Data = %a, Length = %d, ImageSize = %a,  
Format = %d, GpiH = %d  
%Data = %b%b %b%b %b%b %b%b %b%b  
ImageSize->cx = %d, ImageSize->cy = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1601 (0X0641)

<u>Description</u>	GpiPop Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIPOP
<u>Minor Code</u>	1601 (0X0641)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Count = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1602 (0X0642)

<u>Description</u>	GpiPtVisible Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIPTVISIBLE
<u>Minor Code</u>	1602 (0X0642)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%Point = %a, GpiH = %d Point->x = %d, Point->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1603 (0X0643)

<u>Description</u>	GpiRectVisible Pre-Invocation
<u>Tracepoint</u>	



Public symbol defined dynamic tracepoint: PMGPI.GPIRECTVISIBLE

**Minor Code**

1603 (0X0643)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Rect = %a, GpiH = %d

Rect->xLeft = %d, Rect->yBottom = %d

Rect->xRight= %d, Rect->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1604 (0X0644)

**Description**

GpiComment Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPICOMMENT

**Minor Code**

1604 (0X0644)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Comment = %a, Count = %d, GpiH = %d

Comment = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1700 (0X06A4)

**Description**

GpiDeleteSetId Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETESETID

**Minor Code**

1700 (0X06A4)

**Trace Groups**

No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %Lcid = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1701 (0X06A5)

**Description**                      GpiQueryNumberSetIds Pre-Invocation

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYNUMBERSETIDS

**Minor Code**                      1701 (0X06A5)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1702 (0X06A6)

**Description**                      GpiQuerySetIds Pre-Invocation

**Tracepoint**                      Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYSETIDS

**Minor Code**                      1702 (0X06A6)

**Trace Groups**                      No groups assigned.

**Trace Types**                      No types assigned.

**Traced Parameters**

   %Count = %d, GpiH = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 1703 (0X06A7)

<u>Description</u>	GpiLoadFonts Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPILOADFONTS
<u>Minor Code</u>	1703 (0X06A7)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<div>%Filename = %a, GpiH = %d Filename = %s</div>

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1704 (0X06A8)

<u>Description</u>	GpiUnloadFonts Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIUNLOADFONTS
<u>Minor Code</u>	1704 (0X06A8)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	<div>%Filename = %a, GpiH = %d Filename = %s</div>

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1705 (0X06A9)

<u>Description</u>	GpiCreateLogFont Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPICREATELOGFONT
<u>Minor Code</u>	

1705 (0X06A9)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Lcid = %d, FontName = %a, GpiH = %d

FontName = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1706 (0X06AA)

**Description**

GpiQueryFonts Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYFONTS

**Minor Code**

1706 (0X06AA)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Length = %d, FontsReq = %a, FaceName = %a,

Options = %d, GpiH = %d

FontsReg = %d

FaceName = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1707 (0X06AB)

**Description**

GpiQueryFontMetrics Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYFONTMETRICS

**Minor Code**

1707 (0X06AB)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Length = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1708 (0X06AC)

**Description**

GpiQueryKerningPairs Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYKERNINGPAIRS

**Minor Code**

1708 (0X06AC)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%KerningPairs = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1709 (0X06AD)

**Description**

GpiQueryWidthTable Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYWIDTHTABLE

**Minor Code**

1709 (0X06AD)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Count = %d, FirstChar = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1710 (0X06AE)

<b><u>Description</u></b>	GpiSetCp Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETCP
<b><u>Minor Code</u></b>	1710 (0X06AE)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%CodePage = %w, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1711 (0X06AF)

<b><u>Description</u></b>	GpiQueryCp Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCP
<b><u>Minor Code</u></b>	1711 (0X06AF)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1712 (0X06B0)

<b><u>Description</u></b>	GpiQueryFontFileDescriptions Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYFONTFILEDESCRIPTIONS
<b><u>Minor Code</u></b>	1712 (0X06B0)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Filename = %a, GpiH = %d

Filename = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1800 (0X0708)

**Description**

GpiDeleteBitmap Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEBITMAP

**Minor Code**

1800 (0X0708)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HBitmap = %a, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1801 (0X0709)

**Description**

GpiSetBitmap Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAP

**Minor Code**

1801 (0X0709)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HBitmap = %a, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1802 (0X070A)

<b><u>Description</u></b>	GpiBitBlt Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIBITBLT
<b><u>Minor Code</u></b>	1802 (0X070A)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Options = %d, Rop = %d, Pointl = %a, Count = %d, GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 1803 (0X070B)

<b><u>Description</u></b>	GpiWCBitBlt Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIWCBITBLT
<b><u>Minor Code</u></b>	1803 (0X070B)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Options = %d, Rop = %d, Pointl = %a, Count = %d, GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 1804 (0X070C)

<b><u>Description</u></b>	GpiCreateBitmap Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPICREATEBITMAP
<b><u>Minor Code</u></b>	



1804 (0X070C)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Data = %a, Options = %d, GpiH = %d

Data = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1805 (0X070D)

**Description**

GpiSetBitmapDimension Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAPDIMENSION

**Minor Code**

1805 (0X070D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Hbitmap = %d, SizeL = %a

SizeL->cx = %d, SizeL->cy = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1806 (0X070E)

**Description**

GpiQueryBitmapDimension Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPDIMENSION

**Minor Code**

1806 (0X070E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Hbitmap = %a

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1807 (0X070F)

### Description

GpiQueryDeviceBitmapFormats Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYDEVICEBITMAPFORMATS

### Minor Code

1807 (0X070F)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Count = %a, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1808 (0X0710)

### Description

GpiQueryBitmapParameters Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPPARAMETERS

### Minor Code

1808 (0X0710)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%HBitmap = %a

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1809 (0X0711)

### Description

GpiQueryBitmapBits Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPBITS
<b><u>Minor Code</u></b>	1809 (0X0711)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Long = %d, Long = %d, GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 1810 (0X0712)

<b><u>Description</u></b>	GpiSetBitmapBits Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAPBITS
<b><u>Minor Code</u></b>	1810 (0X0712)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Data = %a, Scans = %d, ScanStart = %d, GpiH = %d Data = %s

## PMGPI Major Code: 0X00C5 Minor Code: 1811 (0X0713)

<b><u>Description</u></b>	GpiSetPel Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETPEL
<b><u>Minor Code</u></b>	1811 (0X0713)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.

#### Traced Parameters

%Point = %a, GpiH = %d

%Point->x = %d, Point->y

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1812 (0X0714)

#### Description

GpiQueryPel Pre-Invocation

#### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYPEL

#### Minor Code

1812 (0X0714)

#### Trace Groups

No groups assigned.

#### Trace Types

No types assigned.

#### Traced Parameters

%HPS = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1813 (0X0715)

#### Description

GpiSetBitmapId Pre-Invocation

#### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETBITMAPID

#### Minor Code

1813 (0X0715)

#### Trace Groups

No groups assigned.

#### Trace Types

No types assigned.

#### Traced Parameters

%Lcid = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1814 (0X0716)

<b><u>Description</u></b>	GpiQueryBitmapHandle Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYBITMAPHANDLE
<b><u>Minor Code</u></b>	1814 (0X0716)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Lcid = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1900 (0X076C)

<b><u>Description</u></b>	GpiCreateRegion Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPICREATEREGION
<b><u>Minor Code</u></b>	1900 (0X076C)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Rct = %a, Count = %d, GpiH = %d Rct->xLeft = %d, Rct->yBottom = %d Rct->xRight = %d, Rct->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1901 (0X076D)

<b><u>Description</u></b>	GpiSetRegion Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPISETREGION
<b><u>Minor Code</u></b>	

1901 (0X076D)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Rct = %a, Count = %d, Hrgn = %a, GpiH = %d

Rct->xLeft = %d, Rct->yBottom = %d

Rct->xRight = %d, Rct->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1902 (0X076E)

**Description**

GpiDestroyRegion Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIDESTROYREGION

**Minor Code**

1902 (0X076E)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Hrgn = %a, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1903 (0X076F)

**Description**

GpiCombineRegion Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPICOMBINEREGION

**Minor Code**

1903 (0X076F)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Mode = %d, Src2 = %a, Src1 = %a, Dest = %a, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1904 (0X0770)

### Description

GpiEqualRegion Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIEQUALREGION

### Minor Code

1904 (0X0770)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Src2 = %a, Src1 = %a, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1905 (0X0771)

### Description

GpiOffsetRegion Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIOFFSETREGION

### Minor Code

1905 (0X0771)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Pointl = %a, Hrgn = %a, GpiH = %d

Pointl->x = %d, Pointl->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1906 (0X0772)

<b><u>Description</u></b>	GpiPtInRegion Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIPTINREGION
<b><u>Minor Code</u></b>	1906 (0X0772)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Pointl = %a, Hrgn = %a, GpiH = %d Pointl->x = %d, Pointl->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1907 (0X0773)

<b><u>Description</u></b>	GpiRectInRegion Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIRECTINREGION
<b><u>Minor Code</u></b>	1907 (0X0773)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Rct = %a, Hrgn = %a, GpiH = %d Rct->xLeft = %d, Rct->yBottom = %d Rct->xRight = %d, Rct->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1908 (0X0774)

<b><u>Description</u></b>	GpiQueryRegionBox Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYREGIONBOX
<b><u>Minor Code</u></b>	1908 (0X0774)



**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Hrgn = %a, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1909 (0X0775)

**Description**

GpiQueryRegionRects Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYREGIONRECTS

**Minor Code**

1909 (0X0775)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Rct = %a, Hrgn = %a, GpiH = %d

Rct->xLeft = %d, Rct->yBottom = %d

Rct->xRight = %d, Rct->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 1910 (0X0776)

**Description**

GpiPaintRegion Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIPAINTREGION

**Minor Code**

1910 (0X0776)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HRgn = %a, GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2000 (0X07D0)

### Description

GpiSetClipRegion Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPISETCLIPREGION

### Minor Code

2000 (0X07D0)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Hrgn = %a, GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2001 (0X07D1)

### Description

GpiQueryClipRegion Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCLIPREGION

### Minor Code

2001 (0X07D1)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Hps = %d, GpiH = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2002 (0X07D2)

### Description

GpiQueryClipBox Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYCLIPBOX
<b><u>Minor Code</u></b>	2002 (0X07D2)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%GpiH = %d

## PMGPI Major Code: 0X00C5 Minor Code: 2003 (0X07D3)

<b><u>Description</u></b>	GpiIntersectClipRectangle Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIINTERSECTCLIPRECTANGLE
<b><u>Minor Code</u></b>	2003 (0X07D3)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Rct = %a, GpiH = %d Rct->xLeft = %d, Rct->yBottom = %d Rct->xRight = %d, Rct->yTop = %d

## PMGPI Major Code: 0X00C5 Minor Code: 2004 (0X07D4)

<b><u>Description</u></b>	GpiExcludeClipRectangle Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIEXCLUDECLIPRECTANGLE
<b><u>Minor Code</u></b>	2004 (0X07D4)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	

No types assigned.

**Traced Parameters**

%Rct = %a, GpiH = %d

Rct->xLeft = %d, Rct->yBottom = %d

Rct->xRight = %d, Rct->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2005 (0X07D5)

**Description**

GpiOffsetClipRegion Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPIOFFSETCLIPREGION

**Minor Code**

2005 (0X07D5)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Point = %a, GpiH = %d

Point->x = %d, Point->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2100 (0X0834)

**Description**

GpiLoadMetaFile Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.GPILOADMETAFILE

**Minor Code**

2100 (0X0834)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Filename = %a, GpiH = %d

Filename = %s

-----

PMGPI Major Code: 0X00C5 Minor Code: 2101 (0X0835)

<u>Description</u>	GpiPlayMetaFile Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPIPLAYMETAFILE
<u>Minor Code</u>	2101 (0X0835)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %Count2 = %d, GpiH = %d OptArray = %a, Count1 = %d OptArray = %s

-----

PMGPI Major Code: 0X00C5 Minor Code: 2102 (0X0836)

<u>Description</u>	GpiSaveMetaFile Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.GPISAVEMETAFILE
<u>Minor Code</u>	2102 (0X0836)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	 %Filename = %a, GpiH = %d Filename = %s

-----

PMGPI Major Code: 0X00C5 Minor Code: 2103 (0X0837)

<b><u>Description</u></b>	GpiDeleteMetaFile Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIDELETEMETAFILE
<b><u>Minor Code</u></b>	2103 (0X0837)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HMF = %d, GpiH = %d

## -----

### PMGPI Major Code: 0X00C5 Minor Code: 2104 (0X0838)

<b><u>Description</u></b>	GpiCopyMetaFile Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPICOPYMETAFILE
<b><u>Minor Code</u></b>	2104 (0X0838)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%HMF = %d, GpiH = %d

## -----

### PMGPI Major Code: 0X00C5 Minor Code: 2105 (0X0839)

<b><u>Description</u></b>	GpiQueryMetaFileLength Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMETAFILELENGTH
<b><u>Minor Code</u></b>	2105 (0X0839)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%HMF = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2106 (0X083A)

**Description**  
GpiQueryMetaFileBits Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPIQUERYMETAFILEBITS

**Minor Code**  
2106 (0X083A)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Length = %d, Offset = %d, GpiH = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2107 (0X083B)

**Description**  
GpiSetMetaFileBits Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.GPISETMETAFILEBITS

**Minor Code**  
2107 (0X083B)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Buffer = %a, Length = %d, Offset = %d, GpiH = %d  
Buffer = %s

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2300 (0X08FC)

<u><b>Description</b></u>	MTIDPStoreMeta Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMGPI.MTIDPSTOREMETA
<u><b>Minor Code</b></u>	2300 (0X08FC)
<u><b>Trace Groups</b></u>	No groups assigned.
<u><b>Trace Types</b></u>	No types assigned.
<u><b>Traced Parameters</b></u>	%Long = %d, Word = %w, Pbyte = %a, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2301 (0X08FD)

<u><b>Description</b></u>	MTEnableKerningMeta Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMGPI.MTENABLEKERNINGMETA
<u><b>Minor Code</b></u>	2301 (0X08FD)
<u><b>Trace Groups</b></u>	No groups assigned.
<u><b>Trace Types</b></u>	No types assigned.
<u><b>Traced Parameters</b></u>	%Long = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2302 (0X08FE)

<u><b>Description</b></u>	MTDisplayFlagMeta Pre-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMGPI.MTDISPLAYFLAGMETA
<u><b>Minor Code</b></u>	2302 (0X08FE)
<u><b>Trace Groups</b></u>	No groups assigned.



**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%Bool = %d, HDC = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 2303 (0X08FF)

**Description**  
MTCreateLogColorTableMeta Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.MTCREATELOGCOLORTABLEMETA

**Minor Code**  
2303 (0X08FF)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%ULong = %d,ULong = %d,ULong = %d,ULong = %d,HDC = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 2304 (0X0900)

**Description**  
MTSetCodePageMeta Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMGPI.MTSETCODEPAGEMETA

**Minor Code**  
2304 (0X0900)

**Trace Groups**  
No groups assigned.

**Trace Types**  
No types assigned.

**Traced Parameters**  
  
%ULong = %d, HDC = %d

-----

PMGPI Major Code: 0X00C5 Minor Code: 2305 (0X0901)

<b><u>Description</u></b>	MTDeleteSetIDMeta Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.MTDELETESETIDMETA
<b><u>Minor Code</u></b>	2305 (0X0901)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%ULong = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2306 (0X0902)

<b><u>Description</u></b>	MTSetGraphicsFieldMeta Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.MTSETGRAPHICSFIELDMETA
<b><u>Minor Code</u></b>	2306 (0X0902)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Rect = %a, HDC = %d Rect->xLeft = %d, Rect->yBottom = %d Rect->xRight = %d, Rect->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2307 (0X0903)

<b><u>Description</u></b>	MTResetMeta Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.MTRESETMETA
<b><u>Minor Code</u></b>	

2307 (0X0903)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ULong = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2308 (0X0904)

**Description**

MTEraseMeta Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTERASEMETA

**Minor Code**

2308 (0X0904)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2309 (0X0905)

**Description**

MTAssociateMeta Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTASSOCIATEMETA

**Minor Code**

2309 (0X0905)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HDC = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2310 (0X0906)

**Description**

MTVerifyPageUnits Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTVERIFYPAGEUNITS

**Minor Code**

2310 (0X0906)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ULong = %d, HDC = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2311 (0X0907)

**Description**

MTBitBlitMeta Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTBITBLTMETA

**Minor Code**

2311 (0X0907)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Long = %d, Long = %d, Pointl = %a, HDC1= %d, HDC2= %d

Pointl->x = %d, Pointl->y = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2312 (0X0908)

**Description**

MTWCBitBlitMeta Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTWCBITBLTMETA

**Minor Code**

2312 (0X0908)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HBITMAP = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2313 (0X0909)

**Description**

MTSetPelMeta Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTSETPELMETA

**Minor Code**

2313 (0X0909)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Pointl = %a, HDC = %d

Pointl->x = %d, Pointl->y = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2314 (0X090A)

**Description**

MTSelectClipMeta Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTSELECTCLIPMETA

**Minor Code**

2314 (0X090A)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%Hrgn = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2315 (0X090B)

### Description

MTClipRegionMeta Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.MTCLIPREGIONMETA

### Minor Code

2315 (0X090B)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%UShort = %w, Rect = %a, HDC = %d

Rect->xLeft = %d, Rect->yBottom = %d

Rect->xRight = %d, Rect->yTop = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2316 (0X090C)

### Description

MTPaintRegionMeta Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMGPI.MTPAINTREGIONMETA

### Minor Code

2316 (0X090C)

### Trace Groups

No groups assigned.

### Trace Types

No types assigned.

### Traced Parameters

%Hrgn = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2317 (0X090D)

<b><u>Description</u></b>	MTDevEscape Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.MTDEVESCAPE
<b><u>Minor Code</u></b>	2317 (0X090D)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Long = %d, Long = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2318 (0X090E)

<b><u>Description</u></b>	MTRestorePS Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.MTRESTOREPS
<b><u>Minor Code</u></b>	2318 (0X090E)
<b><u>Trace Groups</u></b>	No groups assigned.
<b><u>Trace Types</u></b>	No types assigned.
<b><u>Traced Parameters</u></b>	%Long = %d, HDC = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2319 (0X090F)

<b><u>Description</u></b>	MTSavePS Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMGPI.MTSAVEPS
<b><u>Minor Code</u></b>	2319 (0X090F)
<b><u>Trace Groups</u></b>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HDC = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2320 (0X0910)

**Description**

MTStartReadRequest Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTSTARTREADREQUEST

**Minor Code**

2320 (0X0910)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HMF = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2321 (0X0911)

**Description**

MTEndReadRequest Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMGPI.MTENDREADREQUEST

**Minor Code**

2321 (0X0911)

**Trace Groups**

No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%HMF = %d

---

## PMGPI Major Code: 0X00C5 Minor Code: 2322 (0X0912)



<u>Description</u>	MtStartWriteRequest Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.MTSTARTWRITEREQUEST
<u>Minor Code</u>	2322 (0X0912)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%HMF = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2323 (0X0913)

<u>Description</u>	MtEndWriteRequest Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.MTENDWRITEREQUEST
<u>Minor Code</u>	2323 (0X0913)
<u>Trace Groups</u>	No groups assigned.
<u>Trace Types</u>	No types assigned.
<u>Traced Parameters</u>	%HMF = %d

-----

## PMGPI Major Code: 0X00C5 Minor Code: 2324 (0X0914)

<u>Description</u>	MtLongByteSwap Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMGPI.MTLONGBYTESWAP
<u>Minor Code</u>	2324 (0X0914)
<u>Trace Groups</u>	No groups assigned.

**Trace Types**

No types assigned.

**Traced Parameters**

%ULong = %d

-----

# PMSPL.DLL Trace Events

The tracepoints for the PMSPL.DLL major code are identified in the following table. These tracepoints are dynamic tracepoints.

**Delay:**

Some of the trace information tables in this document contain large amounts of data and may take several seconds to display.

[Trace events for PMSPL Major Code: 0X00C6, sorted by minor code.](#)  
[Trace events for PMSPL Major Code: 0X00C6 ,sorted by tracepoint.](#)

-----

## Trace Events for PMSPL Major Code: 0X00C6, Sorted by Minor Code

- 00002 (0X0002) SplControlDevice Pre-Invocation
- 00003 (0X0003) SplQueryDevice Pre-Invocation
- 00004 (0X0004) SplEnumDevice Pre-Invocation
- 00005 (0X0005) SplCreateDevice Pre-Invocation
- 00006 (0X0006) SplSetDevice Pre-Invocation
- 00007 (0X0007) SplDeleteDevice Pre-Invocation
- 00008 (0X0008) SplReleaseJob Pre-Invocation
- 00009 (0X0009) SplHoldJob Pre-Invocation
- 00010 (0X000A) SplDeleteJob Pre-Invocation
- 00011 (0X000B) DosPrintJobSchedule Pre-Invocation
- 00012 (0X000C) DosPrintJobAdd Pre-Invocation
- 00013 (0X000D) DosPrintJobAdd2 Pre-Invocation
- 00014 (0X000E) SplQueryJob Pre-Invocation
- 00015 (0X000F) SplSetJob Pre-Invocation
- 00016 (0X0010) SplEnumJob Pre-Invocation
- 00017 (0X0011) SplCreateQueue Pre-Invocation
- 00018 (0X0012) SplHoldQueue Pre-Invocation
- 00019 (0X0013) SplReleaseQueue Pre-Invocation
- 00020 (0X0014) SplDeleteQueue Pre-Invocation
- 00021 (0X0015) SplPurgeQueue Pre-Invocation
- 00022 (0X0016) SplQueryQueue Pre-Invocation
- 00023 (0X0017) SplSetQueue Pre-Invocation
- 00024 (0X0018) SplEnumQueue Pre-Invocation
- 00025 (0X0019) SplEnumDriver Pre-Invocation
- 00026 (0X001A) SplEnumQueueProcessor Pre-Invocation
- 00027 (0X001B) SplEnumPort Pre-Invocation
- 00028 (0X001C) DosPrintJobGetId Pre-Invocation
- 00029 (0X001D) Spl32PrmSpool Pre-Invocation
- 00030 (0X001E) SplQueryDriver Pre-Invocation
- 00031 (0X001F) SplSetDriver Pre-Invocation
- 00032 (0X0020) SplCopyJob Pre-Invocation
- 00033 (0X0021) SplQueryJobFile Pre-Invocation
- 00034 (0X0022) SplEnumPrinter SplEnumQueue Pre-Invocation
- 00049 (0X0031) Spl32QmOpen Pre-Invocation

00050 (0X0032) Spl32QmStartDoc Pre-Invocation  
00051 (0X0033) Spl32QmWrite Pre-Invocation  
00052 (0X0034) Spl32QmWriteFile Pre-Invocation  
00053 (0X0035) Spl32QmEndDoc Pre-Invocation  
00054 (0X0036) Spl32QmAbortDoc Pre-Invocation  
00055 (0X0037) Spl32QmClose Pre-Invocation  
00056 (0X0038) Spl32QmAbort Pre-Invocation  
00057 (0X0039) Spl32QmQueryPinfo Pre-Invocation  
00058 (0X003A) Spl32QmSetStatus Pre-Invocation  
00059 (0X003B) Spl32QmSetup Pre-Invocation  
00081 (0X0051) Spl32MessageBox Pre-Invocation  
00082 (0X0052) Prt32Open Pre-Invocation  
00083 (0X0053) Prt32Write Pre-Invocation  
00084 (0X0054) Prt32DevIOCtl Pre-Invocation  
00085 (0X0055) Prt32Close Pre-Invocation  
00086 (0X0056) Prt32Abort Pre-Invocation  
00087 (0X0057) PrtNewPage Pre-Invocation  
00088 (0X0058) PrtResetAbort Pre-Invocation  
00089 (0X0059) PrtAbortDoc Pre-Invocation  
00113 (0X0071) Spl32StdOpen Pre-Invocation  
00113 (0X0071) Spl32StdOpen Pre-Invocation  
00114 (0X0072) Spl32StdClose Pre-Invocation  
00114 (0X0072) Spl32StdClose Pre-Invocation  
00115 (0X0073) Spl32StdStart Pre-Invocation  
00116 (0X0074) Spl32StdStop Pre-Invocation  
00117 (0X0075) Spl32StdQueryLength Pre-Invocation  
00118 (0X0076) Spl32StdGetBits Pre-Invocation  
00119 (0X0077) Spl32StdDelete Pre-Invocation  
00256 (0X0100) SplFSOpen Pre-Invocation  
00257 (0X0101) SplFSFirstWrite Pre-Invocation  
00258 (0X0102) SplFSWriteFail Pre-Invocation  
00259 (0X0103) SplFSClose Pre-Invocation  
00260 (0X0104) SplFSSetTitle Pre-Invocation  
00261 (0X0105) SplFSActCP Pre-Invocation  
00262 (0X0106) SplFSVerifyCP Pre-Invocation  
00263 (0X0107) SplFSReturnCPAct Pre-Invocation  
00264 (0X0108) AttachPort Pre-Invocation  
00265 (0X0109) DetachPort Pre-Invocation  
00304 (0X0130) PrintDestControl Pre-Invocation  
00305 (0X0131) PrintDestGetInfo Pre-Invocation  
00306 (0X0132) PrintDestEnum Pre-Invocation  
00307 (0X0133) PrintDestAdd Pre-Invocation  
00308 (0X0134) PrintDestSetInfo Pre-Invocation  
00309 (0X0135) PrintDestDel Pre-Invocation  
00320 (0X0140) PrintJobContinue Pre-Invocation  
00321 (0X0141) PrintJobPause Pre-Invocation  
00322 (0X0142) PrintJobDel Pre-Invocation  
00323 (0X0143) PrintJobSchedule Pre-Invocation  
00324 (0X0144) PrintJobAdd Pre-Invocation  
00325 (0X0145) PrintJobGetInfo Pre-Invocation  
00326 (0X0146) PrintJobSetInfo Pre-Invocation  
00327 (0X0147) PrintJobEnum Pre-Invocation  
00336 (0X0150) PrintQPause Pre-Invocation  
00337 (0X0151) PrintQPurge Pre-Invocation  
00338 (0X0152) PrintQContinue Pre-Invocation  
00339 (0X0153) PrintQAdd Pre-Invocation  
00340 (0X0154) PrintQDel Pre-Invocation  
00341 (0X0155) PrintQGetInfo Pre-Invocation  
00342 (0X0156) PrintQSetInfo Pre-Invocation  
00343 (0X0157) PrintQEnum Pre-Invocation  
00344 (0X0158) PrintDriverEnum Pre-Invocation  
00345 (0X0159) PrintQProcessorEnum Pre-Invocation  
00346 (0X015A) PrintPortEnum Pre-Invocation  
00368 (0X0170) SplWarning Pre-Invocation  
00368 (0X0170) SplRWarning Pre-Invocation  
00369 (0X0171) SplError Pre-Invocation  
00369 (0X0171) SplRError Pre-Invocation  
00370 (0X0172) SplPanic Pre-Invocation  
00370 (0X0172) SplRPanic Pre-Invocation  
00371 (0X0173) SplErrNotRunning Pre-Invocation  
00372 (0X0174) SplErrNoMemory Pre-Invocation  
00373 (0X0175) SplLogWarning Pre-Invocation  
00374 (0X0176) SplLogError Pre-Invocation

00384 (0X0180) PostRefreshMsg Pre-Invocation  
32770 (0X8002) SplControlDevice Post-Invocation  
32771 (0X8003) SplQueryDevice Post-Invocation  
32772 (0X8004) SplEnumDevice Post-Invocation  
32773 (0X8005) SplCreateDevice Post-Invocation  
32774 (0X8006) SplSetDevice Post-Invocation  
32775 (0X8007) SplDeleteDevice Post-Invocation  
32776 (0X8008) SplReleaseJob Post-Invocation  
32777 (0X8009) SplHoldJob Post-Invocation  
32778 (0X800A) SplDeleteJob Post-Invocation  
32779 (0X800B) DosPrintJobSchedule Post-Invocation  
32780 (0X800C) DosPrintJobAdd Post-Invocation  
32781 (0X800D) DosPrintJobAdd2 Post-Invocation  
32782 (0X800E) SplQueryJob Post-Invocation  
32783 (0X800F) SplSetJob Post-Invocation  
32784 (0X8010) SplEnumJob Post-Invocation  
32785 (0X8011) SplCreateQueue Post-Invocation  
32786 (0X8012) SplHoldQueue Post-Invocation  
32787 (0X8013) SplReleaseQueue Post-Invocation  
32788 (0X8014) SplDeleteQueue Post-Invocation  
32789 (0X8015) SplPurgeQueue Post-Invocation  
32790 (0X8016) SplQueryQueue Post-Invocation  
32791 (0X8017) SplSetQueue Post-Invocation  
32792 (0X8018) SplEnumQueue Post-Invocation  
32793 (0X8019) SplEnumDriver Post-Invocation  
32794 (0X801A) SplEnumQueueProcessor Post-Invocation  
32795 (0X801B) SplEnumPort Post-Invocation  
32796 (0X801C) DosPrintJobGetId Post-Invocation  
32797 (0X801D) Spl32PrmSpool Post-Invocation  
32798 (0X801E) SplQueryDriver Post-Invocation  
32799 (0X801F) SplSetDriver Post-Invocation  
32800 (0X8020) SplCopyJob Post-Invocation  
32801 (0X8021) SplQueryJobFile Post-Invocation  
32802 (0X8022) SplEnumPrinter Post-Invocation  
32817 (0X8031) Spl32QmOpen Post-Invocation  
32818 (0X8032) Spl32QmStartDoc Post-Invocation  
32819 (0X8033) Spl32QmWrite Post-Invocation  
32820 (0X8034) Spl32QmWriteFile Post-Invocation  
32821 (0X8035) Spl32QmEndDoc Post-Invocation  
32822 (0X8036) Spl32QmAbortDoc Post-Invocation  
32823 (0X8037) Spl32QmClose Post-Invocation  
32824 (0X8038) Spl32QmAbort Post-Invocation  
32825 (0X8039) Spl32QmQueryPinfo Post-Invocation  
32826 (0X803A) Spl32QmSetStatus Post-Invocation  
32827 (0X803B) Spl32QmSetup Post-Invocation  
32849 (0X8051) Spl32MessageBox Post-Invocation  
32850 (0X8052) Prt32Open Post-Invocation  
32851 (0X8053) Prt32Write Post-Invocation  
32852 (0X8054) Prt32DevIOCtl Post-Invocation  
32853 (0X8055) Prt32Close Post-Invocation  
32854 (0X8056) Prt32Abort Post-Invocation  
32855 (0X8057) PrtNewPage Post-Invocation  
32856 (0X8058) PrtResetAbort Post-Invocation  
32857 (0X8059) PrtAbortDoc Post-Invocation  
32881 (0X8071) Spl32StdOpen Post-Invocation  
32882 (0X8072) Spl32StdClose Post-Invocation  
32883 (0X8073) Spl32StdStart Post-Invocation  
32884 (0X8074) Spl32StdStop Post-Invocation  
32885 (0X8075) Spl32StdQueryLength Post-Invocation  
32886 (0X8076) Spl32StdGetBits Post-Invocation  
32887 (0X8077) Spl32StdDelete Post-Invocation  
33024 (0X8100) SplFSOpen Post-Invocation  
33025 (0X8101) SplFSFirstWrite Post-Invocation  
33026 (0X8102) SplFSWriteFail Post-Invocation  
33027 (0X8103) SplFSClose Post-Invocation  
33028 (0X8104) SplFSSetTitle Post-Invocation  
33029 (0X8105) SplFSActCP Post-Invocation  
33030 (0X8106) SplFSVerifyCP Post-Invocation  
33031 (0X8107) SplFSReturnCPAct Post-Invocation  
33032 (0X8108) AttachPort Post-Invocation  
33033 (0X8109) DetachPort Post-Invocation  
33034 (0X810A) GetNextId Post-Invocation  
33072 (0X8130) PrintDestControl Post-Invocation

33073 (0X8131) PrintDestGetInfo Post-Invocation  
33074 (0X8132) PrintDestEnum Post-Invocation  
33075 (0X8133) PrintDestAdd Post-Invocation  
33076 (0X8134) PrintDestSetInfo Post-Invocation  
33077 (0X8135) PrintDestDel Post-Invocation  
33088 (0X8140) PrintJobContinue Post-Invocation  
33089 (0X8141) PrintJobPause Post-Invocation  
33090 (0X8142) PrintJobDel Post-Invocation  
33091 (0X8143) PrintJobSchedule Post-Invocation  
33092 (0X8144) PrintJobAdd Post-Invocation  
33093 (0X8145) PrintJobGetInfo Post-Invocation  
33094 (0X8146) PrintJobSetInfo Post-Invocation  
33095 (0X8147) PrintJobEnum Post-Invocation  
33104 (0X8150) PrintQPause Post-Invocation  
33105 (0X8151) PrintQPurge Post-Invocation  
33106 (0X8152) PrintQContinue Post-Invocation  
33107 (0X8153) PrintQAdd Post-Invocation  
33108 (0X8154) PrintQDel Post-Invocation  
33109 (0X8155) PrintQGetInfo Post-Invocation  
33110 (0X8156) PrintQSetInfo Post-Invocation  
33111 (0X8157) PrintQEnum Post-Invocation  
33112 (0X8158) PrintDriverEnum Post-Invocation  
33113 (0X8159) PrintQProcessorEnum Post-Invocation  
33114 (0X815A) PrintPortEnum Post-Invocation

---

## Trace Events for PMSPL Major Code: 0X00C6, Sorted by Tracepoint

AttachPort 00264 (0X0108)  
AttachPort 33032 (0X8108)  
DetachPort 00265 (0X0109)  
DetachPort 33033 (0X8109)  
DosPrintJobAdd 00012 (0X000C)  
DosPrintJobAdd 32780 (0X800C)  
DosPrintJobAdd2 00013 (0X000D)  
DosPrintJobAdd2 32781 (0X800D)  
DosPrintJobGetId 00028 (0X001C)  
DosPrintJobGetId 32796 (0X801C)  
DosPrintJobSchedule 00011 (0X000B)  
DosPrintJobSchedule 32779 (0X800B)  
GetNextId 33034 (0X810A)  
PRT32OPEN 00082 (0X0052)  
PRT32WRITE 00083 (0X0053)  
PostRefreshMsg 00384 (0X0180)  
PrintDestAdd 00307 (0X0133)  
PrintDestAdd 33075 (0X8133)  
PrintDestControl 00304 (0X0130)  
PrintDestControl 33072 (0X8130)  
PrintDestDel 00309 (0X0135)  
PrintDestDel 33077 (0X8135)  
PrintDestEnum 00306 (0X0132)  
PrintDestEnum 33074 (0X8132)  
PrintDestGetInfo 00305 (0X0131)  
PrintDestGetInfo 33073 (0X8131)  
PrintDestSetInfo 00308 (0X0134)  
PrintDestSetInfo 33076 (0X8134)  
PrintDriverEnum 00344 (0X0158)  
PrintDriverEnum 33112 (0X8158)  
PrintJobAdd 00324 (0X0144)  
PrintJobAdd 33092 (0X8144)  
PrintJobContinue 00320 (0X0140)  
PrintJobContinue 33088 (0X8140)  
PrintJobDel 00322 (0X0142)  
PrintJobDel 33090 (0X8142)

PrintJobEnum 00327 (0X0147)  
PrintJobEnum 33095 (0X8147)  
PrintJobGetInfo 00325 (0X0145)  
PrintJobGetInfo 33093 (0X8145)  
PrintJobPause 00321 (0X0141)  
PrintJobPause 33089 (0X8141)  
PrintJobSchedule 00323 (0X0143)  
PrintJobSchedule 33091 (0X8143)  
PrintJobSetInfo 00326 (0X0146)  
PrintJobSetInfo 33094 (0X8146)  
PrintPortEnum 00346 (0X015A)  
PrintPortEnum 33114 (0X815A)  
PrintQAdd 00339 (0X0153)  
PrintQAdd 33107 (0X8153)  
PrintQContinue 00338 (0X0152)  
PrintQContinue 33106 (0X8152)  
PrintQDel 00340 (0X0154)  
PrintQDel 33108 (0X8154)  
PrintQEnum 00343 (0X0157)  
PrintQEnum 33111 (0X8157)  
PrintQGetInfo 00341 (0X0155)  
PrintQGetInfo 33109 (0X8155)  
PrintQPause 00336 (0X0150)  
PrintQPause 33104 (0X8150)  
PrintQProcessorEnum 00345 (0X0159)  
PrintQProcessorEnum 33113 (0X8159)  
PrintQPurge 00337 (0X0151)  
PrintQPurge 33105 (0X8151)  
PrintQSetInfo 00342 (0X0156)  
PrintQSetInfo 33110 (0X8156)  
Prt32Abort 00086 (0X0056)  
Prt32Abort 32854 (0X8056)  
Prt32Close 00085 (0X0055)  
Prt32Close 32853 (0X8055)  
Prt32DevIOCtl 00084 (0X0054)  
Prt32DevIOCtl 32852 (0X8054)  
Prt32Open 32850 (0X8052)  
Prt32Write 32851 (0X8053)  
PrtAbortDoc 00089 (0X0059)  
PrtAbortDoc 32857 (0X8059)  
PrtNewPage 00087 (0X0057)  
PrtNewPage 32855 (0X8057)  
PrtResetAbort 00088 (0X0058)  
PrtResetAbort 32856 (0X8058)  
SPL32STDCLOSE 00114 (0X0072)  
SPL32STDDELETE 00119 (0X0077)  
SPL32STDOPEN 00113 (0X0071)  
SPL32STDQUERYLENGTH 00117 (0X0075)  
SPL32STDSTART 00115 (0X0073)  
SPL32STDSTOP 00116 (0X0074)  
Spl32MessageBox 00081 (0X0051)  
Spl32MessageBox 32849 (0X8051)  
Spl32PrmSpool 00029 (0X001D)  
Spl32PrmSpool 32797 (0X801D)  
Spl32QmAbort 00056 (0X0038)  
Spl32QmAbort 32824 (0X8038)  
Spl32QmAbortDoc 00054 (0X0036)  
Spl32QmAbortDoc 32822 (0X8036)  
Spl32QmClose 00055 (0X0037)  
Spl32QmClose 32823 (0X8037)  
Spl32QmEndDoc 00053 (0X0035)  
Spl32QmEndDoc 32821 (0X8035)  
Spl32QmOpen 00049 (0X0031)  
Spl32QmOpen 32817 (0X8031)  
Spl32QmQueryPinfo 00057 (0X0039)  
Spl32QmQueryPinfo 32825 (0X8039)  
Spl32QmSetStatus 00058 (0X003A)  
Spl32QmSetStatus 32826 (0X803A)  
Spl32QmSetup 00059 (0X003B)  
Spl32QmSetup 32827 (0X803B)  
Spl32QmStartDoc 00050 (0X0032)  
Spl32QmStartDoc 32818 (0X8032)  
Spl32QmWrite 00051 (0X0033)

Spl32QmWrite 32819 (0X8033)  
Spl32QmWriteFile 00052 (0X0034)  
Spl32QmWriteFile 32820 (0X8034)  
Spl32StdClose 00114 (0X0072)  
Spl32StdClose 32882 (0X8072)  
Spl32StdDelete 32887 (0X8077)  
Spl32StdGetBits 00118 (0X0076)  
Spl32StdGetBits 32886 (0X8076)  
Spl32StdOpen 00113 (0X0071)  
Spl32StdOpen 32881 (0X8071)  
Spl32StdQueryLength 32885 (0X8075)  
Spl32StdStart 32883 (0X8073)  
Spl32StdStop 32884 (0X8074)  
SplControlDevice 00002 (0X0002)  
SplControlDevice 32770 (0X8002)  
SplCopyJob 00032 (0X0020)  
SplCopyJob 32800 (0X8020)  
SplCreateDevice 00005 (0X0005)  
SplCreateDevice 32773 (0X8005)  
SplCreateQueue 00017 (0X0011)  
SplCreateQueue 32785 (0X8011)  
SplDeleteDevice 00007 (0X0007)  
SplDeleteDevice 32775 (0X8007)  
SplDeleteJob 00010 (0X000A)  
SplDeleteJob 32778 (0X800A)  
SplDeleteQueue 00020 (0X0014)  
SplDeleteQueue 32788 (0X8014)  
SplEnumDevice 00004 (0X0004)  
SplEnumDevice 32772 (0X8004)  
SplEnumDriver 00025 (0X0019)  
SplEnumDriver 32793 (0X8019)  
SplEnumJob 00016 (0X0010)  
SplEnumJob 32784 (0X8010)  
SplEnumPort 00027 (0X001B)  
SplEnumPort 32795 (0X801B)  
SplEnumPrinter 00034 (0X0022)  
SplEnumPrinter 32802 (0X8022)  
SplEnumQueue 00024 (0X0018)  
SplEnumQueue 32792 (0X8018)  
SplEnumQueueProcessor 00026 (0X001A)  
SplEnumQueueProcessor 32794 (0X801A)  
SplErrNoMemory 00372 (0X0174)  
SplErrNotRunning 00371 (0X0173)  
SplError 00369 (0X0171)  
SplFSActCP 00261 (0X0105)  
SplFSActCP 33029 (0X8105)  
SplFSClose 00259 (0X0103)  
SplFSClose 33027 (0X8103)  
SplFSFirstWrite 00257 (0X0101)  
SplFSFirstWrite 33025 (0X8101)  
SplFSOpen 00256 (0X0100)  
SplFSOpen 33024 (0X8100)  
SplFSReturnCPAct 00263 (0X0107)  
SplFSReturnCPAct 33031 (0X8107)  
SplFSSetTitle 00260 (0X0104)  
SplFSSetTitle 33028 (0X8104)  
SplFSVerifyCP 00262 (0X0106)  
SplFSVerifyCP 33030 (0X8106)  
SplFSWriteFail 00258 (0X0102)  
SplFSWriteFail 33026 (0X8102)  
SplHoldJob 00009 (0X0009)  
SplHoldJob 32777 (0X8009)  
SplHoldQueue 00018 (0X0012)  
SplHoldQueue 32786 (0X8012)  
SplLogError 00374 (0X0176)  
SplLogWarning 00373 (0X0175)  
SplPanic 00370 (0X0172)  
SplPurgeQueue 00021 (0X0015)  
SplPurgeQueue 32789 (0X8015)  
SplQueryDevice 00003 (0X0003)  
SplQueryDevice 32771 (0X8003)  
SplQueryDriver 00030 (0X001E)  
SplQueryDriver 32798 (0X801E)

SplQueryJob 00014 (0X000E)  
SplQueryJob 32782 (0X800E)  
SplQueryJobFile 00033 (0X0021)  
SplQueryJobFile 32801 (0X8021)  
SplQueryQueue 00022 (0X0016)  
SplQueryQueue 32790 (0X8016)  
SplRError 00369 (0X0171)  
SplRPanic 00370 (0X0172)  
SplRWarning 00368 ( 0X0170)  
SplReleaseJob 00008 (0X0008)  
SplReleaseJob 32776 (0X8008)  
SplReleaseQueue 00019 (0X0013)  
SplReleaseQueue 32787 (0X8013)  
SplSetDevice 00006 (0X0006)  
SplSetDevice 32774 (0X8006)  
SplSetDriver 00031 (0X001F)  
SplSetDriver 32799 (0X801F)  
SplSetJob 00015 (0X000F)  
SplSetJob 32783 (0X800F)  
SplSetQueue 00023 (0X0017)  
SplSetQueue 32791 (0X8017)  
SplWarning 00368 (0X0170)

-----

## PMSPL Major Code: 0X00C6 Minor Code: 2 (0X0002)

### Description

SplControlDevice Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplControlDevice

### Minor Code

2 (0X0002)

### Trace Groups

DOS

### Trace Types

PRE

### Traced Parameters

ulControl=%F, PortName=%s, ComputerName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 3 (0X0003)

### Description

SplQueryDevice Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplQueryDevice

### Minor Code

3 (0X0003)

### Trace Groups

DOS



<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ulLevel=%F, cbBuf=%F, pszPrintDeviceName=%s ComputerName=%s
-----	

## PMSPL Major Code: 0X00C6 Minor Code: 4 (0X0004)

<u>Description</u>	SplEnumDevice Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplEnumDevice
<u>Minor Code</u>	4 (0X0004)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ulLevel=%F, cbBuf=%F, ComputerName=%s
-----	

## PMSPL Major Code: 0X00C6 Minor Code: 5 (0X0005)

<u>Description</u>	SplCreateDevice Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplCreateDevice
<u>Minor Code</u>	5 (0X0005)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ulLevel=%F, cbBuf=%F, ComputerName=%s
-----	

# PMSPL Major Code: 0X00C6 Minor Code: 6 (0X0006)

Description	SplSetDevice Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSPL.SplSetDevice
Minor Code	6 (0X0006)
Trace Groups	DOS
Trace Types	PRE
Traced Parameters	pszPrinter=%s, ulParmNum=%F ulLevel=%F, cbBuf=%F ComputerName=%s

# PMSPL Major Code: 0X00C6 Minor Code: 7 (0X0007)

Description	SplDeleteDevice Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSPL.SplDeleteDevice
Minor Code	7 (0X0007)
Trace Groups	DOS
Trace Types	PRE
Traced Parameters	pszPrinter=%s, ComputerName=%s

# PMSPL Major Code: 0X00C6 Minor Code: 8 (0X0008)

Description	SplReleaseJob Pre-Invocation
Tracepoint	Public symbol defined dynamic tracepoint: PMSPL.SplReleaseJob

**Minor Code** 8 (0X0008)

**Trace Groups** DOS

**Trace Types** PRE

**Traced Parameters**

ulJob=%F, ComputerName=%s, QueueName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 9 (0X0009)

**Description** SplHoldJob Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplHoldJob

**Minor Code** 9 (0X0009)

**Trace Groups** DOS

**Trace Types** PRE

**Traced Parameters**

ulJob=%F, ComputerName=%s, QueueName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 10 (0X000A)

**Description** SplDeleteJob Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplDeleteJob

**Minor Code** 10 (0X000A)

**Trace Groups** DOS

**Trace Types** PRE

**Traced Parameters**

ulJob=%F, ComputerName=%s, QueueName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 11 (0X000B)

<u>Description</u>	DosPrintJobSchedule Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobSchedule
<u>Minor Code</u>	11 (0X000B)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ulJob=%w, ComputerName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 12 (0X000C)

<u>Description</u>	DosPrintJobAdd Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd
<u>Minor Code</u>	12 (0X000C)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	QueueName=%s, cbBuf=%w, pBuf=%r%F ComputerName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 13 (0X000D)

<u>Description</u>	DosPrintJobAdd2 Pre-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd2

**Minor Code**

13 (0X000D)

**Trace Groups**

DOS

**Trace Types**

PRE

**Traced Parameters**

QueueName=%s, ulLevel=%w, cbBuf=%w, pBuf=%r%F

ComputerName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 14 (0X000E)

**Description**

SplQueryJob Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplQueryJob

**Minor Code**

14 (0X000E)

**Trace Groups**

DOS

**Trace Types**

PRE

**Traced Parameters**

ulJob=%F, ulLevel=%F, cbBuf=%F

ComputerName=%s, QueueName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 15 (0X000F)

**Description**

SplSetJob Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplSetJob

**Minor Code**

15 (0X000F)

**Trace Groups**

DOS

**Trace Types**

PRE

**Traced Parameters**

ulJob=%F, ulParmNum=%F ulLevel=%F, cbBuf=%F, pBuf=%u  
ComputerName=%s QueueName=%s

-----

PMSPL Major Code: 0X00C6 Minor Code: 16 (0X0010)

**Description**

SplEnumJob Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplEnumJob

**Minor Code**

16 (0X0010)

**Trace Groups**

DOS

**Trace Types**

PRE

**Traced Parameters**

QueueName=%s, ulLevel=%F, cbBuf=%F, ComputerName=%s

-----

PMSPL Major Code: 0X00C6 Minor Code: 17 (0X0011)

**Description**

SplCreateQueue Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplCreateQueue

**Minor Code**

17 (0X0011)

**Trace Groups**

DOS

**Trace Types**

PRE

**Traced Parameters**

ulLevel=%F, cbBuf=%F, pbBuf=%r%F  
ComputerName=%s

-----

PMSPL Major Code: 0X00C6 Minor Code: 18 (0X0012)

<b><u>Description</u></b>	SplHoldQueue Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplHoldQueue
<b><u>Minor Code</u></b>	18 (0X0012)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	QueueName=%s, ComputerName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 19 (0X0013)

<b><u>Description</u></b>	SplReleaseQueue Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplReleaseQueue
<b><u>Minor Code</u></b>	19 (0X0013)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	QueueName=%s, ComputerName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 20 (0X0014)

<b><u>Description</u></b>	SplDeleteQueue Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplDeleteQueue
<b><u>Minor Code</u></b>	20 (0X0014)
<b><u>Trace Groups</u></b>	DOS

<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	QueueName=%s, ComputerName=%s
	-----

## PMSPL Major Code: 0X00C6 Minor Code: 21 (0X0015)

<u>Description</u>	SplPurgeQueue Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplPurgeQueue
<u>Minor Code</u>	21 (0X0015)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	QueueName=%s, ComputerName=%s
	-----

## PMSPL Major Code: 0X00C6 Minor Code: 22 (0X0016)

<u>Description</u>	SplQueryQueue Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplQueryQueue
<u>Minor Code</u>	22 (0X0016)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	QueueName=%s, ulLevel=%F, cbBuf=%F, ComputerName=%s
	-----

## PMSPL Major Code: 0X00C6 Minor Code: 23 (0X0017)



<u>Description</u>	SplSetQueue Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplSetQueue
<u>Minor Code</u>	23 (0X0017)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	QueueName=%s, ulParmNum=%F ulLevel=%F, cbBuf=%F, pBuf=%r%F ComputerName=%s

## PMSPL Major Code: 0X00C6 Minor Code: 24 (0X0018)

<u>Description</u>	SplEnumQueue Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueue
<u>Minor Code</u>	24 (0X0018)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ulLevel=%F, cbBuf=%F, ComputerName=%s

## PMSPL Major Code: 0X00C6 Minor Code: 25 (0X0019)

<u>Description</u>	SplEnumDriver Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplEnumDriver
<u>Minor Code</u>	25 (0X0019)

**Trace Groups** DOS

**Trace Types** PRE

**Traced Parameters**

ulLevel=%F, cbBuf=%F, ComputerName=%s

-----

PMSPL Major Code: 0X00C6 Minor Code: 26 (0X001A)

**Description** SplEnumQueueProcessor Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueueProcessor

**Minor Code** 26 (0X001A)

**Trace Groups** DOS

**Trace Types** PRE

**Traced Parameters**

ulLevel=%F, cbBuf=%F, ComputerName=%s

-----

PMSPL Major Code: 0X00C6 Minor Code: 27 (0X001B)

**Description** SplEnumPort Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplEnumPort

**Minor Code** 27 (0X001B)

**Trace Groups** DOS

**Trace Types** PRE

**Traced Parameters**

ulLevel=%F, cbBuf=%F, ComputerName=%s

-----

# PMSPL Major Code: 0X00C6 Minor Code: 28 (0X001C)

<u>Description</u>	DosPrintJobGetId Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobGetId
<u>Minor Code</u>	28 (0X001C)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hFile=%F, cbInfo=%F

# PMSPL Major Code: 0X00C6 Minor Code: 29 (0X001D)

<u>Description</u>	Spl32PrmSpool Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32PrmSpool
<u>Minor Code</u>	29 (0X001D)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	pszInLogAddr=%s, pszOutLogAddr=%s

# PMSPL Major Code: 0X00C6 Minor Code: 30 (0X001E)

<u>Description</u>	SplQueryDriver Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplQueryDriver
<u>Minor Code</u>	30 (0X001E)

<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ulLevel=%F, cbBuf=%F, DriverName=%s, PrinterName=%s ComputerName=%s
-----	

## PMSPL Major Code: 0X00C6 Minor Code: 31 (0X001F)

<u>Description</u>	SplSetDriver Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplSetDriver
<u>Minor Code</u>	31 (0X001F)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	ulLevel=%F, ulParmNum=%F, cbBuf=%F DriverName=%s, PrinterName=%s, ComputerName=%s
-----	

## PMSPL Major Code: 0X00C6 Minor Code: 32 (0X0020)

<u>Description</u>	SplCopyJob Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplCopyJob
<u>Minor Code</u>	32 (0X0020)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

SourceJobID=%F, srcQueue=%s, srcComputer=%s

TargetQueue=%s, TargetComputer=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33 (0X0021)

### Description

SplQueryJobFile Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplQueryJobFile

### Minor Code

33 (0X0021)

### Trace Groups

DOS

### Trace Types

PRE

### Traced Parameters

JobID=%F, Queue=%s, Computer=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 34 (0X0022)

### Description

SplEnumPrinter SplEnumQueue Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplEnumPrinter

### Minor Code

34 (0X0022)

### Trace Groups

DOS

### Trace Types

PRE

### Traced Parameters

uLevel=%F, flType=%F, cbBuf=%F, ComputerName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 49 (0X0031)

### Description

	Spl32QmOpen Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmOpen
<b><u>Minor Code</u></b>	49 (0X0031)
<b><u>Trace Groups</u></b>	SPLQM
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>cData=%F pQMOPENDATA=%F pszLogAddress=%F</p> <p>pszDriverName=%F pdrv=%F pszDataType=%F</p> <p>pszComment=%F pszQueueProcName=%F pszQProcParams=%F</p> <p>pszSpoolerParams=%F pszNetworkParams=%F</p> <p>LogAddr-%s DriverName-%s DataType-%s</p> <p>pdrv-&gt;cb=%F pdrv-&gt;lVersion=%F pdrv-&gt;szDeviceName-%s</p> <p>SpoolerParams-%s Comment-%s QProc-%s QProcParms-%s</p>

## PMSPL Major Code: 0X00C6 Minor Code: 50 (0X0032)

<b><u>Description</u></b>	Spl32QmStartDoc Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmStartDoc
<b><u>Minor Code</u></b>	50 (0X0032)
<b><u>Trace Groups</u></b>	SPLQM
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	<p>hSpl=%F, pszDocument=%s</p>

## PMSPL Major Code: 0X00C6 Minor Code: 51 (0X0033)

<b><u>Description</u></b>	Spl32QmWrite Pre-Invocation
---------------------------	-----------------------------

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWrite
<u>Minor Code</u>	51 (0X0033)
<u>Trace Groups</u>	SPLQM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hSpl =%F cbDataIn=%F pbData=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 52 (0X0034)

<u>Description</u>	Spl32QmWriteFile Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWriteFile
<u>Minor Code</u>	52 (0X0034)
<u>Trace Groups</u>	SPLQM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hSpl=%F Filename to write=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 53 (0X0035)

<u>Description</u>	Spl32QmEndDoc Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmEndDoc
<u>Minor Code</u>	53 (0X0035)
<u>Trace Groups</u>	SPLQM
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

hSpl =%F

---

## PMSPL Major Code: 0X00C6 Minor Code: 54 (0X0036)

**Description**

Spl32QmAbortDoc Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbortDoc

**Minor Code**

54 (0X0036)

**Trace Groups**

SPLQM

**Trace Types**

PRE

**Traced Parameters**

hSpl =%F

---

## PMSPL Major Code: 0X00C6 Minor Code: 55 (0X0037)

**Description**

Spl32QmClose Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmClose

**Minor Code**

55 (0X0037)

**Trace Groups**

SPLQM

**Trace Types**

PRE

**Traced Parameters**

hSpl =%F

---

## PMSPL Major Code: 0X00C6 Minor Code: 56 (0X0038)

**Description**

Spl32QmAbort Pre-Invocation



<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbort
<b><u>Minor Code</u></b>	56 (0X0038)
<b><u>Trace Groups</u></b>	SPLQM
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	hSpl =%F

## PMSPL Major Code: 0X00C6 Minor Code: 57 (0X0039)

<b><u>Description</u></b>	Spl32QmQueryPinfo Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmQueryPinfo
<b><u>Minor Code</u></b>	57 (0X0039)
<b><u>Trace Groups</u></b>	SPLQM
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	
	Logical address=%s

## PMSPL Major Code: 0X00C6 Minor Code: 58 (0X003A)

<b><u>Description</u></b>	Spl32QmSetStatus Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetStatus
<b><u>Minor Code</u></b>	58 (0X003A)
<b><u>Trace Groups</u></b>	SPLQM
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	

Job ID=%w, NewStatus=%w, Logical address=%s, pszStatus=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 59 (0X003B)

### Description

Spl32QmSetup Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetup

### Minor Code

59 (0X003B)

### Trace Groups

SPLQM

### Trace Types

PRE

### Traced Parameters

Type=%F DopData=%F %F %F %F

LogAddr=%s DriverName=%s DataType=%s

pdriv->cb=%F pdriv->lVersion=%F pdriv->szDeviceName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 81 (0X0051)

### Description

Spl32MessageBox Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.Spl32MessageBox

### Minor Code

81 (0X0051)

### Trace Groups

PRT

### Trace Types

PRE

### Traced Parameters

fErrInfo=%F, fErrData=%F, fStyle=%F, LogAddr=%s

Caption=%s, Text=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 82 (0X0052)

<u>Description</u>	Prt32Open Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PRT32OPEN
<u>Minor Code</u>	82 (0X0052)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	pszDeviceName=%s, openFlag=%F, openMode=%F, reserved=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 83 (0X0053)

<u>Description</u>	Prt32Write Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PRT32WRITE
<u>Minor Code</u>	83 (0X0053)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hFile=%F, pchData=%F, cbData=%F, *pchData=%r%b

-----

## PMSPL Major Code: 0X00C6 Minor Code: 84 (0X0054)

<u>Description</u>	Prt32DevIOCtl Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Prt32DevIOCtl
<u>Minor Code</u>	84 (0X0054)
<u>Trace Groups</u>	PRT

**Trace Types**  
PRE

**Traced Parameters**  
  
hFile=%F, uFunction=%F, uCategory=%F  
\*pchParm=%F, \*pchData=%F

PMSPL Major Code: 0X00C6 Minor Code: 85 (0X0055)

**Description**  
Prt32Close Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.Prt32Close

**Minor Code**  
85 (0X0055)

**Trace Groups**  
PRT

**Trace Types**  
PRE

**Traced Parameters**  
  
hFile=%F

PMSPL Major Code: 0X00C6 Minor Code: 86 (0X0056)

**Description**  
Prt32Abort Pre-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.Prt32Abort

**Minor Code**  
86 (0X0056)

**Trace Groups**  
PRT

**Trace Types**  
PRE

**Traced Parameters**  
  
hFile=%F

# PMSPL Major Code: 0X00C6 Minor Code: 87 (0X0057)

<u>Description</u>	PrtNewPage Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrtNewPage
<u>Minor Code</u>	87 (0X0057)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hFile=%F ulPageNumber=%F

-----

# PMSPL Major Code: 0X00C6 Minor Code: 88 (0X0058)

<u>Description</u>	PrtResetAbort Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrtResetAbort
<u>Minor Code</u>	88 (0X0058)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hFile=%F

-----

# PMSPL Major Code: 0X00C6 Minor Code: 89 (0X0059)

<u>Description</u>	PrtAbortDoc Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrtAbortDoc
<u>Minor Code</u>	89 (0X0059)

**Trace Groups**

PRT

**Trace Types**

PRE

**Traced Parameters**

hFile=%F, pchData=%F, cbData=%F, ulFlags=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 113 (0X0071)

**Description**

Spl32StdOpen Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SPL32STDOPEN

**Minor Code**

113 (0X0071)

**Trace Groups**

STD

**Trace Types**

PRE

**Traced Parameters**

hDC=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 113 (0X0071)

**Description**

Spl32StdOpen Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.Spl32StdOpen

**Minor Code**

113 (0X0071)

**Trace Groups**

STD

**Trace Types**

PRE

**Traced Parameters**

hDC=%F

-----

# PMSPL Major Code: 0X00C6 Minor Code: 114 (0X0072)

<u>Description</u>	Spl32StdClose Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SPL32STDCLOSE
<u>Minor Code</u>	114 (0X0072)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hDC=%F

# PMSPL Major Code: 0X00C6 Minor Code: 114 (0X0072)

<u>Description</u>	Spl32StdClose Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdClose
<u>Minor Code</u>	114 (0X0072)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hDC=%F

# PMSPL Major Code: 0X00C6 Minor Code: 115 (0X0073)

<u>Description</u>	Spl32StdStart Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SPL32STDSTART
<u>Minor Code</u>	

115 (0X0073)

**Trace Groups**

STD

**Trace Types**

PRE

**Traced Parameters**

hDC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 116 (0X0074)

**Description**

Spl32StdStop Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SPL32STDSTOP

**Minor Code**

116 (0X0074)

**Trace Groups**

STD

**Trace Types**

PRE

**Traced Parameters**

hDC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 117 (0X0075)

**Description**

Spl32StdQueryLength Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SPL32STDQUERYLENGTH

**Minor Code**

117 (0X0075)

**Trace Groups**

STD

**Trace Types**

PRE

**Traced Parameters**

hMetaFile=%F



-----

## PMSPL Major Code: 0X00C6 Minor Code: 118 (0X0076)

<u>Description</u>	Spl32StdGetBits Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdGetBits
<u>Minor Code</u>	118 (0X0076)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hMetaFile=%F, offData=%F, cbData=%F, pchData=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 119 (0X0077)

<u>Description</u>	Spl32StdDelete Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SPL32STDDELETE
<u>Minor Code</u>	119 (0X0077)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	hMetaFile=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 256 (0X0100)

<u>Description</u>	SplFSDOpen Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSDOpen

**Minor Code** 256 (0X0100)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**

uHandle=%w Codepage=%w, Key=%F, Port=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 257 (0X0101)

**Description** SplFSFirstWrite Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplFSFirstWrite

**Minor Code** 257 (0X0101)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**

uHandle=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 258 (0X0102)

**Description** SplFSWriteFail Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplFSWriteFail

**Minor Code** 258 (0X0102)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**

uHandle=%w, ErrorCode=%w

---

## PMSPL Major Code: 0X00C6 Minor Code: 259 (0X0103)

<u>Description</u>	SplFSClose Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSClose
<u>Minor Code</u>	259 (0X0103)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	uHandle=%w

---

## PMSPL Major Code: 0X00C6 Minor Code: 260 (0X0104)

<u>Description</u>	SplFSSetTitle Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSSetTitle
<u>Minor Code</u>	260 (0X0104)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	uHandle=%w, Title=%s

---

## PMSPL Major Code: 0X00C6 Minor Code: 261 (0X0105)

<u>Description</u>	SplFSActCP Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSActCP

**Minor Code** 261 (0X0105)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**  
uHandle=%w, CodePage=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 262 (0X0106)

**Description** SplFSVerifyCP Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplFSVerifyCP

**Minor Code** 262 (0X0106)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**  
uHandle=%w, CodePage=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 263 (0X0107)

**Description** SplFSReturnCPAct Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplFSReturnCPAct

**Minor Code** 263 (0X0107)

**Trace Groups** FS

**Trace Types** PRE

**Traced Parameters**  
uHandle=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 264 (0X0108)

<u>Description</u>	AttachPort Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.AttachPort
<u>Minor Code</u>	264 (0X0108)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	pPort=%F, PortName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 265 (0X0109)

<u>Description</u>	DetachPort Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DetachPort
<u>Minor Code</u>	265 (0X0109)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	pPort=%F, PortName=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 304 (0X0130)

<u>Description</u>	PrintDestControl Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintDestControl

**Minor Code** 304 (0X0130)

**Trace Groups** PRINTX

**Trace Types** PRE

**Traced Parameters**

pszDest=%s, Control=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 305 (0X0131)

**Description** PrintDestGetInfo Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintDestGetInfo

**Minor Code** 305 (0X0131)

**Trace Groups** PRINTX

**Trace Types** PRE

**Traced Parameters**

pszName=%s, level=%F, cbBuf=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 306 (0X0132)

**Description** PrintDestEnum Pre-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintDestEnum

**Minor Code** 306 (0X0132)

**Trace Groups** PRINTX

**Trace Types** PRE

**Traced Parameters**

level=%F, cbBuf=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 307 (0X0133)

<u>Description</u>	PrintDestAdd Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintDestAdd
<u>Minor Code</u>	307 (0X0133)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	level%F, Buf=%r%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 308 (0X0134)

<u>Description</u>	PrintDestSetInfo Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintDestSetInfo
<u>Minor Code</u>	308 (0X0134)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	pszPrinter=%s, level=%F, cbBuf=%F, parmnum=%F Buf=%r%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 309 (0X0135)

<u>Description</u>	PrintDestDel Pre-Invocation
<u>Tracepoint</u>	

Public symbol defined dynamic tracepoint: PMSPL.PrintDestDel

**Minor Code**

309 (0X0135)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

pszPrinter=%s, level=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 320 (0X0140)

**Description**

PrintJobContinue Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintJobContinue

**Minor Code**

320 (0X0140)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

Job ID=%F, User Name=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 321 (0X0141)

**Description**

PrintJobPause Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintJobPause

**Minor Code**

321 (0X0141)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**



Job ID=%F, User Name=%s

---

## PMSPL Major Code: 0X00C6 Minor Code: 322 (0X0142)

**Description**

PrintJobDel Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintJobDel

**Minor Code**

322 (0X0142)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

Job ID=%F, User Name=%s

---

## PMSPL Major Code: 0X00C6 Minor Code: 323 (0X0143)

**Description**

PrintJobSchedule Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintJobSchedule

**Minor Code**

323 (0X0143)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

Job ID=%w

---

## PMSPL Major Code: 0X00C6 Minor Code: 324 (0X0144)

**Description**

PrintJobAdd Pre-Invocation

<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.PrintJobAdd
<b><u>Minor Code</u></b>	324 (0X0144)
<b><u>Trace Groups</u></b>	PRINTX
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	level=%w, QueueName=%s

## PMSPL Major Code: 0X00C6 Minor Code: 325 (0X0145)

<b><u>Description</u></b>	PrintJobGetInfo Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.PrintJobGetInfo
<b><u>Minor Code</u></b>	325 (0X0145)
<b><u>Trace Groups</u></b>	PRINTX
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Job ID=%F, level=%F, cbBuf=%F

## PMSPL Major Code: 0X00C6 Minor Code: 326 (0X0146)

<b><u>Description</u></b>	PrintJobSetInfo Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.PrintJobSetInfo
<b><u>Minor Code</u></b>	326 (0X0146)
<b><u>Trace Groups</u></b>	PRINTX
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	

Job ID=%F, level=%F, cbBuf=%F, parmnum=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 327 (0X0147)

### Description

PrintJobEnum Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintJobEnum

### Minor Code

327 (0X0147)

### Trace Groups

PRINTX

### Trace Types

PRE

### Traced Parameters

level=%F, cbBuf=%F, Qname=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 336 (0X0150)

### Description

PrintQPause Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintQPause

### Minor Code

336 (0X0150)

### Trace Groups

PRINTX

### Trace Types

PRE

### Traced Parameters

Qname=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 337 (0X0151)

### Description

PrintQPurge Pre-Invocation

<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQPurge
<u>Minor Code</u>	337 (0X0151)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Qname=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 338 (0X0152)

<u>Description</u>	PrintQContinue Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQContinue
<u>Minor Code</u>	338 (0X0152)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	Qname=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 339 (0X0153)

<u>Description</u>	PrintQAdd Pre-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQAdd
<u>Minor Code</u>	339 (0X0153)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	PRE
<u>Traced Parameters</u>	

level=%F, Buf=%r%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 340 (0X0154)

### Description

PrintQDel Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintQDel

### Minor Code

340 (0X0154)

### Trace Groups

PRINTX

### Trace Types

PRE

### Traced Parameters

Qname=%s, level=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 341 (0X0155)

### Description

PrintQGetInfo Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintQGetInfo

### Minor Code

341 (0X0155)

### Trace Groups

PRINTX

### Trace Types

PRE

### Traced Parameters

Qname=%s, level=%F, cbBuf=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 342 (0X0156)

### Description

PrintQSetInfo Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintQSetInfo

**Minor Code**

342 (0X0156)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

Qname=%s, level=%F, cbBuf=%F, parmnum=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 343 (0X0157)

**Description**

PrintQEnum Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintQEnum

**Minor Code**

343 (0X0157)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

level=%F, cbBuf=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 344 (0X0158)

**Description**

PrintDriverEnum Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintDriverEnum

**Minor Code**

344 (0X0158)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

level=%F, cbBuf=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 345 (0X0159)

**Description**

PrintQProcessorEnum Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintQProcessorEnum

**Minor Code**

345 (0X0159)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

level=%F, cbBuf=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 346 (0X015A)

**Description**

PrintPortEnum Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.PrintPortEnum

**Minor Code**

346 (0X015A)

**Trace Groups**

PRINTX

**Trace Types**

PRE

**Traced Parameters**

level=%F, cbBuf=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 368 (0X0170)

<b><u>Description</u></b>	SplWarning Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplWarning
<b><u>Minor Code</u></b>	368 (0X0170)
<b><u>Trace Groups</u></b>	ERROR
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Function=%S, ReturnCode=%F, Other=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 368 (0X0170)

<b><u>Description</u></b>	SplRWarning Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplRWarning
<b><u>Minor Code</u></b>	368 (0X0170)
<b><u>Trace Groups</u></b>	ERROR
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Return Address=%F, rc=%F, value=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 369 (0X0171)

<b><u>Description</u></b>	SplError Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplError
<b><u>Minor Code</u></b>	369 (0X0171)
<b><u>Trace Groups</u></b>	ERROR
<b><u>Trace Types</u></b>	PRE



**Traced Parameters**

Function-%s, ReturnCode=%F, Other=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 369 (0X0171)

**Description**

SplRError Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplRError

**Minor Code**

369 (0X0171)

**Trace Groups**

ERROR

**Trace Types**

PRE

**Traced Parameters**

Return Address=%F, rc=%F, value=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 370 (0X0172)

**Description**

SplPanic Pre-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplPanic

**Minor Code**

370 (0X0172)

**Trace Groups**

ERROR

**Trace Types**

PRE

**Traced Parameters**

Function-%s, ReturnCode=%F, Other=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 370 (0X0172)

<b><u>Description</u></b>	SplRPanic Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplRPanic
<b><u>Minor Code</u></b>	370 (0X0172)
<b><u>Trace Groups</u></b>	ERROR
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Return Address=%F, rc=%F, value=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 371 (0X0173)

<b><u>Description</u></b>	SplErrNotRunning Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplErrNotRunning
<b><u>Minor Code</u></b>	371 (0X0173)
<b><u>Trace Groups</u></b>	ERROR
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	No parameters traced.

-----

## PMSPL Major Code: 0X00C6 Minor Code: 372 (0X0174)

<b><u>Description</u></b>	SplErrNoMemory Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplErrNoMemory
<b><u>Minor Code</u></b>	372 (0X0174)
<b><u>Trace Groups</u></b>	ERROR
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	

DosError=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 373 (0X0175)

### Description

SplLogWarning Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplLogWarning

### Minor Code

373 (0X0175)

### Trace Groups

ERROR

### Trace Types

PRE

### Traced Parameters

PLError=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 374 (0X0176)

### Description

SplLogError Pre-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.SplLogError

### Minor Code

374 (0X0176)

### Trace Groups

ERROR

### Trace Types

PRE

### Traced Parameters

PLError=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 384 (0X0180)

### Description

	PostRefreshMsg Pre-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.PostRefreshMsg
<b><u>Minor Code</u></b>	384 (0X0180)
<b><u>Trace Groups</u></b>	REFRESH
<b><u>Trace Types</u></b>	PRE
<b><u>Traced Parameters</u></b>	Type=%F, JobID=%F, Queue=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32770 (0X8002)

<b><u>Description</u></b>	SplControlDevice Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplControlDevice
<b><u>Minor Code</u></b>	32770 (0X8002)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32771 (0X8003)

<b><u>Description</u></b>	SplQueryDevice Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplQueryDevice
<b><u>Minor Code</u></b>	32771 (0X8003)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	POST

**Traced Parameters**

rc=%F, cbNeeded=%F, pBuf=%r%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32772 (0X8004)

**Description**

SplEnumDevice Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplEnumDevice

**Minor Code**

32772 (0X8004)

**Trace Groups**

DOS

**Trace Types**

POST

**Traced Parameters**

rc=%F, cbNeeded=%F, cReturned=%F, cTotal=%F

pBuf=%r%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32773 (0X8005)

**Description**

SplCreateDevice Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.SplCreateDevice

**Minor Code**

32773 (0X8005)

**Trace Groups**

DOS

**Trace Types**

POST

**Traced Parameters**

rc=%F, pBuf=%r%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32774 (0X8006)

<b><u>Description</u></b>	SplSetDevice Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplSetDevice
<b><u>Minor Code</u></b>	32774 (0X8006)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	rc=%F, pBuf=%r%F

## PMSPL Major Code: 0X00C6 Minor Code: 32775 (0X8007)

<b><u>Description</u></b>	SplDeleteDevice Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplDeleteDevice
<b><u>Minor Code</u></b>	32775 (0X8007)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	rc=%F

## PMSPL Major Code: 0X00C6 Minor Code: 32776 (0X8008)

<b><u>Description</u></b>	SplReleaseJob Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplReleaseJob
<b><u>Minor Code</u></b>	32776 (0X8008)
<b><u>Trace Groups</u></b>	DOS

**Trace Types** POST

**Traced Parameters**  
  
rc=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32777 (0X8009)

**Description** SplHoldJob Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplHoldJob

**Minor Code** 32777 (0X8009)

**Trace Groups** DOS

**Trace Types** POST

**Traced Parameters**  
  
rc=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32778 (0X800A)

**Description** SplDeleteJob Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplDeleteJob

**Minor Code** 32778 (0X800A)

**Trace Groups** DOS

**Trace Types** POST

**Traced Parameters**  
  
rc=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32779 (0X800B)

<u>Description</u>	DosPrintJobSchedule Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobSchedule
<u>Minor Code</u>	32779 (0X800B)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32780 (0X800C)

<u>Description</u>	DosPrintJobAdd Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd
<u>Minor Code</u>	32780 (0X800C)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F, pulJob=%w, pszFileName=%s

-----

PMSPL Major Code: 0X00C6 Minor Code: 32781 (0X800D)

<u>Description</u>	DosPrintJobAdd2 Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobAdd2
<u>Minor Code</u>	32781 (0X800D)
<u>Trace Groups</u>	DOS



**Trace Types** POST

**Traced Parameters**  
rc=%F, pulJob=%w, pszFileName=%s

PMSPL Major Code: 0X00C6 Minor Code: 32782 (0X800E)

**Description** SplQueryJob Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplQueryJob

**Minor Code** 32782 (0X800E)

**Trace Groups** DOS

**Trace Types** POST

**Traced Parameters**  
rc=%F, cbNeeded=%F, pBuffer=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 32783 (0X800F)

**Description** SplSetJob Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplSetJob

**Minor Code** 32783 (0X800F)

**Trace Groups** DOS

**Trace Types** POST

**Traced Parameters**  
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32784 (0X8010)

<u>Description</u>	SplEnumJob Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplEnumJob
<u>Minor Code</u>	32784 (0X8010)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32785 (0X8011)

<u>Description</u>	SplCreateQueue Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplCreateQueue
<u>Minor Code</u>	32785 (0X8011)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32786 (0X8012)

<u>Description</u>	SplHoldQueue Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplHoldQueue
<u>Minor Code</u>	32786 (0X8012)
<u>Trace Groups</u>	DOS

**Trace Types** POST

**Traced Parameters**  
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32787 (0X8013)

**Description** SplReleaseQueue Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplReleaseQueue

**Minor Code** 32787 (0X8013)

**Trace Groups** DOS

**Trace Types** POST

**Traced Parameters**  
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32788 (0X8014)

**Description** SplDeleteQueue Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplDeleteQueue

**Minor Code** 32788 (0X8014)

**Trace Groups** DOS

**Trace Types** POST

**Traced Parameters**  
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32789 (0X8015)

<u>Description</u>	SplPurgeQueue Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplPurgeQueue
<u>Minor Code</u>	32789 (0X8015)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F

## PMSPL Major Code: 0X00C6 Minor Code: 32790 (0X8016)

<u>Description</u>	SplQueryQueue Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplQueryQueue
<u>Minor Code</u>	32790 (0X8016)
<u>Trace Groups</u>	DOS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F, cbNeeded=%F, pBuffer=%r%F

## PMSPL Major Code: 0X00C6 Minor Code: 32791 (0X8017)

<u>Description</u>	SplSetQueue Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplSetQueue
<u>Minor Code</u>	32791 (0X8017)
<u>Trace Groups</u>	DOS

Trace Types  
POST

Traced Parameters  
  
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32792 (0X8018)

Description  
SplEnumQueue Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueue

Minor Code  
32792 (0X8018)

Trace Groups  
DOS

Trace Types  
POST

Traced Parameters  
  
rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pbBuf=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 32793 (0X8019)

Description  
SplEnumDriver Post-Invocation

Tracepoint  
Public symbol defined dynamic tracepoint: PMSPL.SplEnumDriver

Minor Code  
32793 (0X8019)

Trace Groups  
DOS

Trace Types  
POST

Traced Parameters  
  
rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 32794 (0X801A)

<u><b>Description</b></u>	SplEnumQueueProcessor Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSPL.SplEnumQueueProcessor
<u><b>Minor Code</b></u>	32794 (0X801A)
<u><b>Trace Groups</b></u>	DOS
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

## PMSPL Major Code: 0X00C6 Minor Code: 32795 (0X801B)

<u><b>Description</b></u>	SplEnumPort Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSPL.SplEnumPort
<u><b>Minor Code</b></u>	32795 (0X801B)
<u><b>Trace Groups</b></u>	DOS
<u><b>Trace Types</b></u>	POST
<u><b>Traced Parameters</b></u>	rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

## PMSPL Major Code: 0X00C6 Minor Code: 32796 (0X801C)

<u><b>Description</b></u>	DosPrintJobGetId Post-Invocation
<u><b>Tracepoint</b></u>	Public symbol defined dynamic tracepoint: PMSPL.DosPrintJobGetId
<u><b>Minor Code</b></u>	32796 (0X801C)
<u><b>Trace Groups</b></u>	DOS

Trace Types POST

Traced Parameters  
rc=%F, JobID =%F pInfo=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 32797 (0X801D)

Description Spl32PrmSpool Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMSPL.Spl32PrmSpool

Minor Code 32797 (0X801D)

Trace Groups DOS

Trace Types POST

Traced Parameters  
rc=%F

PMSPL Major Code: 0X00C6 Minor Code: 32798 (0X801E)

Description SplQueryDriver Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMSPL.SplQueryDriver

Minor Code 32798 (0X801E)

Trace Groups DOS

Trace Types POST

Traced Parameters  
rc=%F, cbNeeded=%F, pBuffer=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 32799 (0X801F)

<b><u>Description</u></b>	SplSetDriver Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplSetDriver
<b><u>Minor Code</u></b>	32799 (0X801F)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	rc=%F, pBuf=%r%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32800 (0X8020)

<b><u>Description</u></b>	SplCopyJob Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplCopyJob
<b><u>Minor Code</u></b>	32800 (0X8020)
<b><u>Trace Groups</u></b>	DOS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	rc=%F, NewJobID=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32801 (0X8021)

<b><u>Description</u></b>	SplQueryJobFile Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplQueryJobFile
<b><u>Minor Code</u></b>	32801 (0X8021)
<b><u>Trace Groups</u></b>	DOS



**Trace Types**  
POST

**Traced Parameters**  
  
rc=%F, FileName=%s

PMSPL Major Code: 0X00C6 Minor Code: 32802 (0X8022)

**Description**  
SplEnumPrinter Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.SplEnumPrinter

**Minor Code**  
32802 (0X8022)

**Trace Groups**  
DOS

**Trace Types**  
POST

**Traced Parameters**  
  
rc=%F, cReturned=%F, cTotal=%F, cbNeeded=%F, pBuf=%r%F

PMSPL Major Code: 0X00C6 Minor Code: 32817 (0X8031)

**Description**  
Spl32QmOpen Post-Invocation

**Tracepoint**  
Public symbol defined dynamic tracepoint: PMSPL.Spl32QmOpen

**Minor Code**  
32817 (0X8031)

**Trace Groups**  
SPLQM

**Trace Types**  
POST

**Traced Parameters**  
  
Spool File Handle=%F, Possible ErrorCode=%F

PMSPL Major Code: 0X00C6 Minor Code: 32818 (0X8032)

<u>Description</u>	Spl32QmStartDoc Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmStartDoc
<u>Minor Code</u>	32818 (0X8032)
<u>Trace Groups</u>	SPLQM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	fSuccess=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32819 (0X8033)

<u>Description</u>	Spl32QmWrite Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWrite
<u>Minor Code</u>	32819 (0X8033)
<u>Trace Groups</u>	SPLQM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	fSuccess=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32820 (0X8034)

<u>Description</u>	Spl32QmWriteFile Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmWriteFile
<u>Minor Code</u>	32820 (0X8034)
<u>Trace Groups</u>	SPLQM

**Trace Types**

POST

**Traced Parameters**

fSuccess=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32821 (0X8035)

**Description**

Spl32QmEndDoc Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmEndDoc

**Minor Code**

32821 (0X8035)

**Trace Groups**

SPLQM

**Trace Types**

POST

**Traced Parameters**

Job ID=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32822 (0X8036)

**Description**

Spl32QmAbortDoc Post-Invocation

**Tracepoint**

Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbortDoc

**Minor Code**

32822 (0X8036)

**Trace Groups**

SPLQM

**Trace Types**

POST

**Traced Parameters**

fSuccess=%F

-----

PMSPL Major Code: 0X00C6 Minor Code: 32823 (0X8037)

<u>Description</u>	Spl32QmClose Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmClose
<u>Minor Code</u>	32823 (0X8037)
<u>Trace Groups</u>	SPLQM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32824 (0X8038)

<u>Description</u>	Spl32QmAbort Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmAbort
<u>Minor Code</u>	32824 (0X8038)
<u>Trace Groups</u>	SPLQM
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32825 (0X8039)

<u>Description</u>	Spl32QmQueryPinfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32QmQueryPinfo
<u>Minor Code</u>	32825 (0X8039)
<u>Trace Groups</u>	SPLQM

Trace Types POST

Traced Parameters

Job ID=%w

PMSPL Major Code: 0X00C6 Minor Code: 32826 (0X803A)

Description Spl32QmSetStatus Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetStatus

Minor Code 32826 (0X803A)

Trace Groups SPLQM

Trace Types POST

Traced Parameters

fSuccess=%F

PMSPL Major Code: 0X00C6 Minor Code: 32827 (0X803B)

Description Spl32QmSetup Post-Invocation

Tracepoint Public symbol defined dynamic tracepoint: PMSPL.Spl32QmSetup

Minor Code 32827 (0X803B)

Trace Groups SPLQM

Trace Types POST

Traced Parameters

rc=%F Type=%F DopData-%F %F %F %F

LogAddr-%s DriverName-%s DataType-%s

pdriv->cb=%F pdriv->IVersion=%F pdriv->szDeviceName-%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32849 (0X8051)

<u>Description</u>	Spl32MessageBox Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32MessageBox
<u>Minor Code</u>	32849 (0X8051)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32850 (0X8052)

<u>Description</u>	Prt32Open Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Prt32Open
<u>Minor Code</u>	32850 (0X8052)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F, hFile=%F, ActionTaken=%F, Dos Filehandle=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32851 (0X8053)

<u>Description</u>	Prt32Write Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Prt32Write

<u>Minor Code</u>	32851 (0X8053)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	RC=%F, cbWritten=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32852 (0X8054)

<u>Description</u>	Prt32DevIOCtl Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Prt32DevIOCtl
<u>Minor Code</u>	32852 (0X8054)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	RC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32853 (0X8055)

<u>Description</u>	Prt32Close Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Prt32Close
<u>Minor Code</u>	32853 (0X8055)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	RC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32854 (0X8056)

<u>Description</u>	Prt32Abort Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Prt32Abort
<u>Minor Code</u>	32854 (0X8056)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	RC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32855 (0X8057)

<u>Description</u>	PrtNewPage Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrtNewPage
<u>Minor Code</u>	32855 (0X8057)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	RC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32856 (0X8058)

<u>Description</u>	PrtResetAbort Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrtResetAbort



<u>Minor Code</u>	32856 (0X8058)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	RC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32857 (0X8059)

<u>Description</u>	PrtAbortDoc Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrtAbortDoc
<u>Minor Code</u>	32857 (0X8059)
<u>Trace Groups</u>	PRT
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	RC=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32881 (0X8071)

<u>Description</u>	Spl32StdOpen Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdOpen
<u>Minor Code</u>	32881 (0X8071)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	fSuccess=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32882 (0X8072)

<u>Description</u>	Spl32StdClose Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdClose
<u>Minor Code</u>	32882 (0X8072)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	fSuccess=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32883 (0X8073)

<u>Description</u>	Spl32StdStart Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdStart
<u>Minor Code</u>	32883 (0X8073)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	fSuccess=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32884 (0X8074)

<u>Description</u>	Spl32StdStop Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdStop

<b><u>Minor Code</u></b>	32884 (0X8074)
<b><u>Trace Groups</u></b>	STD
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	pBuffer(HSDT)=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32885 (0X8075)

<b><u>Description</u></b>	Spl32StdQueryLength Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdQueryLength
<b><u>Minor Code</u></b>	32885 (0X8075)
<b><u>Trace Groups</u></b>	STD
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Length=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32886 (0X8076)

<b><u>Description</u></b>	Spl32StdGetBits Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdGetBits
<b><u>Minor Code</u></b>	32886 (0X8076)
<b><u>Trace Groups</u></b>	STD
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	fSuccess=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 32887 (0X8077)

<u>Description</u>	Spl32StdDelete Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.Spl32StdDelete
<u>Minor Code</u>	32887 (0X8077)
<u>Trace Groups</u>	STD
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	fSuccess=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33024 (0X8100)

<u>Description</u>	SplFSDOpen Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSDOpen
<u>Minor Code</u>	33024 (0X8100)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%w, JobId=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33025 (0X8101)

<u>Description</u>	SplFSFirstWrite Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSFirstWrite

<b><u>Minor Code</u></b>	33025 (0X8101)
<b><u>Trace Groups</u></b>	FS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Error=%w, CP Activation length=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33026 (0X8102)

<b><u>Description</u></b>	SplFSWriteFail Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplFSWriteFail
<b><u>Minor Code</u></b>	33026 (0X8102)
<b><u>Trace Groups</u></b>	FS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	Error=%w, Response=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33027 (0X8103)

<b><u>Description</u></b>	SplFSClose Post-Invocation
<b><u>Tracepoint</u></b>	Public symbol defined dynamic tracepoint: PMSPL.SplFSClose
<b><u>Minor Code</u></b>	33027 (0X8103)
<b><u>Trace Groups</u></b>	FS
<b><u>Trace Types</u></b>	POST
<b><u>Traced Parameters</u></b>	rc=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33028 (0X8104)

<u>Description</u>	SplFSSetTitle Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSSetTitle
<u>Minor Code</u>	33028 (0X8104)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33029 (0X8105)

<u>Description</u>	SplFSActCP Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSActCP
<u>Minor Code</u>	33029 (0X8105)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33030 (0X8106)

<u>Description</u>	SplFSVerifyCP Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.SplFSVerifyCP

**Minor Code** 33030 (0X8106)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

DosPFSVerifyFont rc=%w

-----

PMSPL Major Code: 0X00C6 Minor Code: 33031 (0X8107)

**Description** SplFSReturnCPAct Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.SplFSReturnCPAct

**Minor Code** 33031 (0X8107)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

Error=%w, CP Activation length=%w

-----

PMSPL Major Code: 0X00C6 Minor Code: 33032 (0X8108)

**Description** AttachPort Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.AttachPort

**Minor Code** 33032 (0X8108)

**Trace Groups** FS

**Trace Types** POST

**Traced Parameters**

rc=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33033 (0X8109)

<u>Description</u>	DetachPort Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.DetachPort
<u>Minor Code</u>	33033 (0X8109)
<u>Trace Groups</u>	FS
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33034 (0X810A)

<u>Description</u>	GetNextId Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.GetNextId
<u>Minor Code</u>	33034 (0X810A)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	Returned Job ID =%w

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33072 (0X8130)

<u>Description</u>	PrintDestControl Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintDestControl



**Minor Code** 33072 (0X8130)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33073 (0X8131)

**Description** PrintDestGetInfo Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintDestGetInfo

**Minor Code** 33073 (0X8131)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F, \*pcbNeeded=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33074 (0X8132)

**Description** PrintDestEnum Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintDestEnum

**Minor Code** 33074 (0X8132)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33075 (0X8133)

<u>Description</u>	PrintDestAdd Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintDestAdd
<u>Minor Code</u>	33075 (0X8133)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33076 (0X8134)

<u>Description</u>	PrintDestSetInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintDestSetInfo
<u>Minor Code</u>	33076 (0X8134)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33077 (0X8135)

<u>Description</u>	PrintDestDel Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintDestDel

**Minor Code** 33077 (0X8135)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33088 (0X8140)

**Description** PrintJobContinue Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintJobContinue

**Minor Code** 33088 (0X8140)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33089 (0X8141)

**Description** PrintJobPause Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintJobPause

**Minor Code** 33089 (0X8141)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33090 (0X8142)

<u>Description</u>	PrintJobDel Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintJobDel
<u>Minor Code</u>	33090 (0X8142)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33091 (0X8143)

<u>Description</u>	PrintJobSchedule Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintJobSchedule
<u>Minor Code</u>	33091 (0X8143)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%W

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33092 (0X8144)

<u>Description</u>	PrintJobAdd Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintJobAdd

**Minor Code** 33092 (0X8144)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**

rc=%w, JobID=%w, Filename=%s

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33093 (0X8145)

**Description** PrintJobGetInfo Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintJobGetInfo

**Minor Code** 33093 (0X8145)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**

rc=%F, cbNeeded=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33094 (0X8146)

**Description** PrintJobSetInfo Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintJobSetInfo

**Minor Code** 33094 (0X8146)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**

rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33095 (0X8147)

<u>Description</u>	PrintJobEnum Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintJobEnum
<u>Minor Code</u>	33095 (0X8147)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33104 (0X8150)

<u>Description</u>	PrintQPause Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQPause
<u>Minor Code</u>	33104 (0X8150)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33105 (0X8151)

<u>Description</u>	PrintQPurge Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQPurge

**Minor Code** 33105 (0X8151)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33106 (0X8152)

**Description** PrintQContinue Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintQContinue

**Minor Code** 33106 (0X8152)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33107 (0X8153)

**Description** PrintQAdd Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintQAdd

**Minor Code** 33107 (0X8153)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33108 (0X8154)

<u>Description</u>	PrintQDel Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQDel
<u>Minor Code</u>	33108 (0X8154)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33109 (0X8155)

<u>Description</u>	PrintQGetInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQGetInfo
<u>Minor Code</u>	33109 (0X8155)
<u>Trace Groups</u>	PRINTX
<u>Trace Types</u>	POST
<u>Traced Parameters</u>	
	rc=%F, cbNeeded=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33110 (0X8156)

<u>Description</u>	PrintQSetInfo Post-Invocation
<u>Tracepoint</u>	Public symbol defined dynamic tracepoint: PMSPL.PrintQSetInfo



**Minor Code** 33110 (0X8156)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33111 (0X8157)

**Description** PrintQEnum Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintQEnum

**Minor Code** 33111 (0X8157)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

-----

## PMSPL Major Code: 0X00C6 Minor Code: 33112 (0X8158)

**Description** PrintDriverEnum Post-Invocation

**Tracepoint** Public symbol defined dynamic tracepoint: PMSPL.PrintDriverEnum

**Minor Code** 33112 (0X8158)

**Trace Groups** PRINTX

**Trace Types** POST

**Traced Parameters**  
rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

---

## PMSPL Major Code: 0X00C6 Minor Code: 33113 (0X8159)

### Description

PrintQProcessorEnum Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintQProcessorEnum

### Minor Code

33113 (0X8159)

### Trace Groups

PRINTX

### Trace Types

POST

### Traced Parameters

rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

---

## PMSPL Major Code: 0X00C6 Minor Code: 33114 (0X815A)

### Description

PrintPortEnum Post-Invocation

### Tracepoint

Public symbol defined dynamic tracepoint: PMSPL.PrintPortEnum

### Minor Code

33114 (0X815A)

### Trace Groups

PRINTX

### Trace Types

POST

### Traced Parameters

rc=%F, cbNeeded=%F, cTotal=%F, cReturned=%F

---

## CONFIG.SYS RAS Statements

OS/2 provides a number of CONFIG.SYS commands and settings specifically for [RAS](#) purposes. Some of these are described in the [OS/2 Command Reference](#), and will not be discussed in detail here. A number of commands were introduced or enhanced in APAR PJ12258, which is applicable to OS/2 2.11. All the commands described here are available with OS/2 Warp 3.0.

The following **CONFIG.SYS** commands comprise the RAS set. Those not completely documented in the OS/2 Command Reference are now discussed in detail.

- `AUTOFAIL` (see [the OS/2 Command Reference](#)).
- `DUMPPROCESS`
- `REIPL`
- `RASKDATA`
- `SCKILLFEATUREENABLED`
- `SHAPIEXCEPTIONHANDLER`
- `SHELLEXCEPTIONHANDLER`
- `SUPPRESSPOPUPS`
- `TRACE` (see [the OS/2 Command Reference](#)). Also see the [System Trace Facility - User Guide](#) chapter in this book.
- `TRACEBUF` (see [the OS/2 Command Reference](#)).
- `TRACEFMT` (see [the OS/2 Command Reference](#)).
- `TRAPDUMP`

-----

## AUTOFAIL

**AUTOFAIL** modifies the processing of media errors. See [the OS/2 Command Reference](#).

-----

## DUMPPROCESS

This command allows the user to activate the process dump facility. When active, any ring 3 (application) process that traps will result in a memory dump being written to a unique dump file. The dump file takes a name of the form *PDUMP.nnn* where *nnn* is an index that is incremented each time a new process dump is created.

The contents of the dump comprise unformatted system and user storage that relates to the trapping process. Included in this are:

- PTDA
- TCB & TSD
- Registers
- Arena records
- MTE and SMTEs
- LDT
- ring 0 stack
- ring 3 stack

### Syntax.

`DUMPPROCESS=x`

### Parameters

*x*

This specifies the drive letter (excluding the colon) to which process dump data sets will be written. These takes the name *PDUMP.nnn* and reside in the root directory of the drive specified. The name and directory cannot be overridden by the user.

**Note:**

See the [Process Dump Formatter](#) section of the Dump Formatter User Guide for information on formatting Process Dumps.

-----

## REIPL

This command allows the user to automate the re-booting (re-IPLing) of the system following an [IPE](#).

**Syntax.**

REIPL=ON | OFF

**Parameters**

**ON**

This specifies the system it to be automatically re-booted following an IPE.

**OFF**

This specified that the system is not to be automatically re-booted following an IPE. The system will remain hung until manually restarted.

**Notes:**

**REIPL** only applied to pre-WARP systems if APAR PJ12258 has been applied.

**REIPL** has no effect when [TRAPDUMP=ON|R0](#) is specified. Whether the system is re-booted following a stand-alone dump is governed by the **OS2DUMP** module. If the the dump is to hard-disk then automatic re-boot occurs, otherwise not.

-----

## RASKDATA

The RASKDATA statement directs the kenel to retain various system control blocks in memory for debugging purposes.

**Syntax.**

RASKDATA=[OTE][,LOCKS | NOLOCKS]

**Parameters**

**OTE**

causes the **OTE**, **STE** and **SMTE** [loader structures](#) to be retained in resident memory for use by the [Dump Formatter .LM command](#)..

The default behaviour is for **OTE** and **SMTE** structure to be allocated from resident memeory, which can causes these to become paged out and thus not present in a system dump.

**LOCKS**

causes long and short term lock records to be retained for use by the [Dump Formatter .MK command](#).

The default behaviour under the **RETAIL** kernel is not to retain lock records.

The default behaviour under the **HSTRICT** and **ALLSTRICT** kernels is to retain lock records.

**LOCKS** and **NOLOCKS** are mutually exclusive options.

#### **NOLOCKS**

causes long and short term lock records not to be retained. If this option is specified then the [Dump Formatter .MK command](#) will not function.

The default behaviour under the **RETAIL** kernel is not to retain lock records.

The default behaviour under the **HSTRICT** and **ALLSTRICT** kernels is to retain lock records.

#### Remarks

Use of **RASKDATA** will impact resident memory requirements may force greater paging activity if insufficient memory is available.

The **OTE** option was introduced with OS/2 Warp V3.0 fix pack 35 and OS/2 Warp V4.0 fix pack 10.

The **LOCKS** and **NOLOCKS** options were introduced with OS/2 Warp V3.0 fix pack 40 and OS/2 Warp V4.0 fix pack 10.

All options are available with OS/2 Warp E-Server

-----

## SCKILLFEATUREENABLED

This command enables the **Kill Feature** of the Warp Centre introduced with OS/2 Warp V4.0.

#### Syntax.

```
SET SCKILLFEATUREENABLED=YES | NO
```

#### Parameters

##### **YES**

Enables the **kill Feature** of the Warp Centre.

##### **NO**

Disables the **kill Feature** of the Warp Centre. This is the default setting.

#### **Note:**

The **Kill Feature** is invoked by clicking on the window list icon of the Warp Centre while holding down the Ctrl key. This causes a list of all executing processes to be displayed. Selecting a process from this list will cause it to be terminated, following a confirmation prompt.

-----

## SHAPIEXCEPTIONHANDLER

This command disables or enables the registration of the exception handler in the **PMSHAPI.DLL** module.

#### Syntax.

```
SET SHAPIEXCEPTIONHANDLER=ON | OFF
```

#### Parameters

## ON

This is the default setting. The shell API DLL exception handler is enabled and normal error recovery takes place whenever a user PM application or the desktop traps.

## OFF

The shell API DLL exception handler is disabled. No additional error recovery provided by the shell takes place when a user application or the desktop traps.

### Notes:

Exception handler registration only occurs during **PMSHAPI.DLL** initialisation. Therefore, a change to the specification of **SHAPIEXCEPTIONHANDLER** will require the system to be rebooted.

The Shell API DLL exception handler will attempt to clean up an application's PM resources.

Under certain circumstances application traps can be pervasive. Either the default error recovery is too efficient to allow the trap to be intercepted or analysed, or the trap recurses to a more serious problem, from which it is also difficult to determine the underlying cause. **SHAPIEXCEPTIONHANDLER** may be used under these circumstances to allow the problem to be intercepted closer to the point of occurrence.

**SHAPIEXCEPTIONHANDLER** may be used with [TRAPDUMP](#) to force a system dump at the point of failure.

Hangs in the shell during initialisation may be the result of a re-cursive trap. **SHAPIEXCEPTIONHANDLER** may be used to intercept this condition.

Since it is difficult to determine whether a potential shell problem involves **PMSHELL.EXE** or **PMSHAPI.DLL** then it is recommended to use **SHAPIEXCEPTIONHANDLER** with [SHELLEXCEPTIONHANDLER](#).

-----

# SHELLEXCEPTIONHANDLER

This command disables or enables the registration of the exception handler in the **PMSHELL.EXE** module.

### Syntax.

```
SET SHELLEXCEPTIONHANDLER=ON|OFF
```

### Parameters

## ON

This is the default setting. The shell's exception handler is enabled and normal error recovery takes place whenever a user PM application or the desktop traps.

## OFF

The shell's exception handler is disabled. No additional error recovery provided by the shell takes place when a user application or the desktop traps.

### Notes:

Exception handler registration only occurs during **PMSHELL.EXE** initialisation. Therefore, a change to the specification of **SHELLEXCEPTIONHANDLER** will require the system to be rebooted.

The Shell's exception handler will attempt to clean up an application's PM resources. In addition if the application is the Desktop (or whatever is specified in [RUNWORKPLACE](#)), then it is restarted.

Under certain circumstances application traps can be pervasive. Either the default error recovery is too efficient to allow the trap to be intercepted or analysed, or the trap recurses to a more serious problem, from which it is also difficult to determine the underlying cause. **SHELLEXCEPTIONHANDLER** may be used under these circumstances to allow the problem to be intercepted closer to the point of occurrence.

**SHELLEXCEPTIONHANDLER** may be used with [TRAPDUMP](#) to force a system dump at the point of failure.

Hangs in the shell during initialisation may be the result of a re-cursive trap. **SHELLEXCEPTIONHANDLER** may be used to

intercept this condition.

Since it is difficult to determine whether a potential shell problem involves **PMSHELL.EXE** or **PMSHAPI.DLL** then it is recommended to use **SHELLEXCEPTIONHANDLER** with [SHAPIEXCEPTIONHANDLER](#).

-----

## SUPPRESSPOPUPS

This command allows the user to suppress the display of trap information pop-up messages and instead direct to trap information to a log data set.

### Syntax.

SUPPRESSPOPUPS=*x*

### Parameters

*x*

This specifies the drive letter (excluding the colon) to which the pop-up log data set will be written. The log takes the name *POPUPLOG.OS2* and resides in the root directory of the drive specified. The name and directory cannot be overridden by the user.

From Fix Pack 29 of OS/2 Warp V3.0 and OS/2 Warp V4.0 GA, trap screen pop-ups are by default logged by default in *POPUPLOG.OS2* in to root directory of the boot drive and not suppressed. *x* may specify **0** to disable automatic logging. In addition, the [TRAPLOG](#) command provides a command line interface to control trap screen logging and suppression independently.

-----

## TRACE

**TRACE** specifies whether tracing of static trace events is to be active from system initialisation or not. See [the OS/2 Command Reference](#) for details. See also the [System Trace Facility - User Guide](#) chapter in this book.

-----

## TRACEBUF

**TRACEBUF** specifies the size of the system trace buffer. See [the OS/2 Command Reference](#) for details.

-----

## TRACEFMT

The **TRACEFMT** utility is used to extract and format the system trace from either a saved trace buffer or the currently active trace buffer. See [the OS/2 Command Reference](#) for details.

-----

## TRAPDUMP

**Warning: Potential Data Loss**

Misuse of this facility may cause loss of vital data. Please read carefully the complete description before use.

The **TRAPDUMP** command controls the stand-alone (system) dump facility of OS/2. It will enable initiation of a stand-alone dump at the instant a ring 3 trap occurs for which no exception handler has intervened.

Ring 0 traps may be also intercepted only on 2.11 systems to which APAR **PJ12258** has been applied (kernel revision 6.624), or on OS/2 Warp.

**Pre-Warp considerations:** The dump process is performed by the hidden module **OS2DUMP**, which resides in the root directory of the boot drive. **OS2DUMP** as supplied with GA versions of OS/2 2.x dumps only to diskette. It may be replaced with a version supplied with OS/2 Problem Determination Package (OS2PDP) which will dump to a hard disk FAT partition that has volume label **SADUMP** or to diskette, depending upon **TRAPDUMP** command specification.

The GA 2.x version of **OS2DUMP** requires the first dump diskette be freshly prepared using the **CREATEDD** command and subsequent diskettes to be formatted. See the on-line OS/2 Command Reference for details of **CREATEDD**.

The OS/2 Problem Determination Package (OS2PDP) version of **OS2DUMP** only requires formatted diskettes, the use of **CREATEDD** being redundant.

When dumping to hard disk the dump partition must to be made known to **TRAPDUMP**. This is done by specifying an optional second parameter.

**OS/2 Warp considerations:** Under OS/2 Warp the **CREATEDD** command is unnecessary and is not distributed with the system. Ordinarily formatted diskettes may be used. Furthermore the enhanced version of **OS2DUMP** that allows dumping to a hard-disk FAT partition is standard. The partition volume label must be **SADUMP**.

#### Syntax.

```
TRAPDUMP=[ ON | OFF | R0 ] [ , PD ] [ , X :
```

#### Parameters

##### **ON**

Specifies that the stand-alone dump process will be automatically initiated whenever an unrecoverable ring 3 trap occurs. For 2.11 systems with APAR PJ12258 or OS/2 Warp, any system **IPE** (including ring 0 traps) will also initiate a dump when **ON** is specified.

##### **OFF**

Specifies that the stand-alone dump process will not initiate automatically when an unrecoverable trap occurs. This is the default option. It does not prohibit the use of the Ctrl-Alt-Numlock-Numlock or Ctrl-Alt-F10-F10 key sequence or the use of [DosForceSystemDump](#) to force a system dump to be initiated.

##### **R0**

Specifies that only ring zero traps and **IPes** will automatically initiate the stand-alone dump process. This option applies only to 2.11 systems with APAR PJ12258 or OS/2 Warp.

##### **Note:**

When an IPE occurs the dump is taken immediately on displaying the IPE trap screen. For the purposes of dump analysis the formatted registers from the IPE screen should be located from the video buffer, which may be viewed using the analyse option from **PMDF**.

##### **PD**

Specifies that a system level process dump will be attempted if the user uses either the Ctrl-Alt-Numlock-Numlock or Ctrl-Alt-F10-F10 key sequences. If for some reason the process dump doesn't complete, a second key sequence will initiate a system dump. On completion of either the system or process dump, the system is re-booted automatically.

##### **X:**

specifies the hard-disk FAT partition to which **OS2DUMP** will write a stand-alone dump. The partition letter must have the colon suffix.

##### **Note:**

The partition may be specified with either ON or OFF. When specified with OFF it will allow a stand-alone dump initiated by Ctrl-Alt-Numlock-Numlock or Ctrl-Alt-F10-F10 key sequences to be written to the dump partition.

Mountable media other than diskette drives are not detectable by OS2DUMP. The letter specifying the dump partition must be



calculated as if any such media were **not** present.

Only hard disk logical drives and primary partitions may be specified.

When dumping to a hard disk partition is selected the system is automatically re-booted on completion of the dump.

System Dump parameter may also be set dynamically from the command line by using the [TRAPDUMP command](#).

### Warnings:

The stand-alone dump process will erase all data on the dump media (disk partition or diskettes) before writing the dump.

Do not specify a disk partition or use diskettes that contain vital data.

---

## Miscellaneous RAS Command Utilities

OS/2 provides a number of command line utilities for use in problem in problem determination and controlling system RAS settings dynamically. The commands described in this section have not been formally documented at all versions of OS/2 and so are described here.

The commands described are:

- [TRAPDUMP](#)
- [TRAPLOG](#)
- [SYSDUMP](#)

---

## TRAPDUMP

The TRAPDUMP command allows the conditions under which a trap will initiate a System Dump to be set dynamically.

### Warning:

The initiation of a System Dump causes an immediate termination of the system without any shutdown. No file system shutdown is performed. The system behaves as if a fatal crash has occurred, thus under rare circumstances data can be lost.

### Syntax.

```
TRAPDUMP [[ON] | [OFF] | [R0]] [x:] [/NOCHECK] [QUERY]
```

### Parameters

<b>ON</b>	enables all application and system traps to initiate a System Dump.
<b>OFF</b>	disables automatic dump initiation.
<b>R0</b>	enables only Ring 0 traps to initiate a System Dump.
<b>x:</b>	specifies the Dump Partition.

## **/NOCHECK**

disable the check for system level OS/2 Warp V4.0. This parameter is now obsolete, since the **TRAPDUMP** command is now available on all supported releases of OS/2.

## **QUERY**

shows the current settings for TRAPDUMP as a CONFIG.SYS statement and an equivalent command line command.

## Remarks

**ON**, **OFF** and **R0** parameters are mutually exclusive.

The **PD** option of the CONFIG.SYS **TRAPDUMP** cannot be set dynamically.

**TRAPDUMP** was made available at fix pack 29 for OS/2 Warp V3.0 and base releases of later versions of OS/2.

---

# TRAPLOG

The TRAPLOG command allows dynamic control of trap and exceptions popup message logging and display.

## Syntax.

```
TRAPLOG [x: | NOLOG] [POPUPS | NOPOPUPS]
```

## Parameters

### **x:**

specifies that trap information is to be logged in x:\POPUPLOG.OS2, x: being any partition drive letter.

### **NOLOG**

disables logging of trap information.

### **POPUPS**

enables the trap information pop-up message (SYS317x).

### **NOPOPUPS**

disables the trap information pop-up message.

## Remarks

If neither **POPUPS** nor **NOPOPUPS** is specified then the state of POPUP suppression is left unchanged.

If neither **x:** nor **NOLOG** is specified then the state of logging is left unchanged.

An applicaiton may temporarily disable exception popup messages by use of the **DosError** API. Use of yhis will not affect logging.

**TRAPLOG** was made available at fix pack 29 for OS/2 Warp V3.0 and base releases of later versions of OS/2.

---

# SYSDUMP

The SYSDUMP command forces a System Dump to be initiated immediately regardless of the TRAPDUMP settings.

## **Warning:**

The initiation of a System Dump causes an immediate termination of the system without any shutdown. No file system shutdown is

performed. The system behaves as if a fatal crash has occurred thus under rare circumstances data can be lost.

#### Syntax.

```
SYSDUMP [ /NOPROMPT]
```

#### Parameters

##### **/NOPROMPT**

The user will not be prompted for a confirmation to proceed with initiating a system dump. The default behaviour is to prompt for confirmation with following message:

```
"Do you want to force a system dump? (Y/N) "
```

#### Remarks

The **SYSDUMP** command may be used from within a CMD file to automate the creation of a system dump.

---

## OS/2 RAS Application Programming Interfaces

This chapter describes the subset of OS/2 **RAS** APIs for use by application programmers, which are not described in the **OS/2 Technical Library, Control Programming Reference**.

#### **CAUTION:**

**Some RAS Programming interfaces may be specific to a particular release of OS/2 or have release specific function.**

The APIs discussed in this section are:

DosSysTrace

DosGetSTDA

DosForceSystemDump

DosDumpProcess

DosSuppressPopUps

DosQueryRASInfo

16-bit Error Logging APIs:

DosLogRegister

DosLogEntry

DosLogRead

32-bit Error Logging APIs:

LogOpen

LogClose

LogAddEntries

---

# DosSysTrace (Static Trace Event Recording)

Static trace recording is available as both an API and a DevHlp routine.

Select one of the following:

- [DosSysTrace](#)
- [DevHlp\\_RAS](#)

---

## DosSysTrace (Add a Trace Record to the System Trace Buffer)

**DosSysTrace** allows a subsystem or system extension to add information to the system trace buffer.

**Note:** **DosSysTrace** is a 16-bit API.

### Coding Examples.

```

        EXTRN    DosSysTrace:FAR

        PUSH     WORD        MajorCode      ; major trace event code (240-255)
        PUSH     WORD        Length         ; length of the variable length
                                                ; area to be recorded (0-512)
        PUSH     WORD        MinorCode      ; minor trace event code (0-255)
        PUSH@    OTHER       Data           ; pointer to the area to be traced
                                                ; (address parameter)

        CALL     DOSSYSTRACE
```

16-bit MASM Example

```

APIRET16 APIENTRY16 DosSysTrace(USHORT MajorCode, USHORT Length,
                                USHORT MinorCode, PCHAR pData);
```

32-bit code Example using CSet/2

### Parameters

MajorCode	The major code to be placed in the trace buffer. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.
Length	The length of the area pointed to by the address parameter. If a length greater than 512 is specified, only 512 bytes will be recorded. If a length of 0 is specified, the address parameter will not be used; however, a dummy doubleword must be pushed on the stack so that all calls use the same stack space.
MinorCode	The minor code to be placed in the trace buffer. This code identifies the specific trace event. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.
pData	The address of the variable length data area which contains additional information that the system trace function will add to the trace buffer. If a length of 0 is specified, the address will not be used, but a value must still be added to the stack.

### Results.

**DosSysTrace** returns the following values:

0  
NO\_ERROR

150  
ERROR\_SYSTEM\_TRACE (Trace is disabled for that event)

```
IF AX = 0
    Data traced
ELSE
    AX = Error_System_Trace
    Data not traced
```

**Note:** An example of when data would not be traced is if the major event code is not currently selected for tracing.

#### Remarks.

All trace records consist of a header and optional data. The header record is built by **DosSysTrace** and contains:

- Major event code
- Minor event code
- Process ID of caller
- Time stamp when the time is different from the previous trace record
- Flag field
- Data field (optional)

The optional data field contains the variable-length data as passed by the caller.

The trace facility maintains an array of 32 bytes (256 bits), in which each bit represents a major event code. This array is updated each time the user enables or disables tracing of a major event. The trace facility checks this array each time it is called to ensure that the major event specified is currently enabled for tracing. The array is located in the [Global Information Segment](#).

A prototype definition for **DosSysTrace** may be found under [RAS API Prototypes](#).

## DevHlp\_RAS (Add a Trace Record to the System Trace Buffer)

The **DevHlp\_RAS** function provides a service for device drivers to add information to the System Trace buffer.

**Note:** DevHlp\_RAS is a 16-bit API.

#### Coding Example.

```
MOV    AX, MajorCode        ; major trace event code (240-255)
MOV    BX, Length           ; length of data area (0-512 bytes)
MOV    CX, MinorCode        ; minor trace event code (0-255)
LDS    SI, pData            ; pointer to trace data
MOV    DL, 28H              ; DevHlp_RAS function code
CALL   [Device_Help]        ; invoke device helper
```

16-bit MASM Example

#### Parameters.

**MajorCode**  
The major code to be placed in the trace buffer. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

**Length**  
The length of the area pointed to by the address parameter. If a length greater than 512 is specified, only 512 bytes

will be recorded. If a length of 0 is specified, the address parameter will not be used; however, a dummy doubleword must be pushed on the stack so that all calls use the same stack space.

#### MinorCode

The minor code to be placed in the trace buffer. This code identifies the specific trace event. Only the low order byte is used. The high order byte should be 0 for future compatibility reasons, but no error checking of the high order byte is performed.

#### pData

The address of the variable length data area which contains additional information that the system trace function will add to the trace buffer. If a length of 0 is specified, the address will not be used, but a value must still be added to the stack.

#### Results.

```
If CF = 0
    Trace record placed in trace buffer
Else
    Data not traced
```

The possible errors are :

- Tracing suspended
- Minor code not being traced
- PID not being traced
- Trace overrun

#### Remarks.

The trace facility maintains an array of 32 bytes (256 bits), in which each bit represents a major event code. This array is updated each time the user enables or disables tracing of a major event. The device driver must check this array before calling **DevHlp\_RAS** to ensure that the major event specified is currently enabled for tracing. This array is located in the [Global Information Segment](#).

All registers are preserved. Interrupts are disabled while the trace data is saved and then re-enabled if they were initially enabled.

-----

## DosGetSTDA (Get The System Trace Data Area)

The **DosGetSTDA** API is a 16 bit API that returns a copy of the system trace buffer (STDA).

#### Syntax

The following 16 bit C language function prototype can be used to call the **DosGetSTDA** API:

```
// 16 bit compiler
extern unsigned far pascal DosGetSTDA( SEL, SHORT, SHORT );

// 32 bit compiler
APIRET16 APIENTRY16 DosGetSTDA( SEL, SHORT, SHORT );

Where:  SEL    is the selector to the private buffer
        SHORT  is the offset to the private buffer
        SHORT  is the size of the buffer
           (maximum value = 64KB)
           records

Returns: 0 - indicates correct operation, buffer is now filled
           with copy of the system trace buffer
        ERROR_SYSTEM_TRACE - System trace is not enabled
```

#### Linker Considerations

In order to successfully resolve **DosGetSTDA** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSGETSTDA=DOSCALL1.119
```

Remarks

DosGetSTDA returns a buffer that contains a copy of the system trace buffer. The buffer is circular with a header record that contains pointers to the first and last data bytes and a pointer to the next byte that was available for writing (the buffer is a snapshot of the system trace buffer at the time that the API was called). A set of trace records follows the header. Each trace record contains a trace event trailer and optionally a timestamp and/or a data field. A Timestamp record is optional and will only exist if bit 2 of the flag field in the Trace Event Trailer is set to OFF.

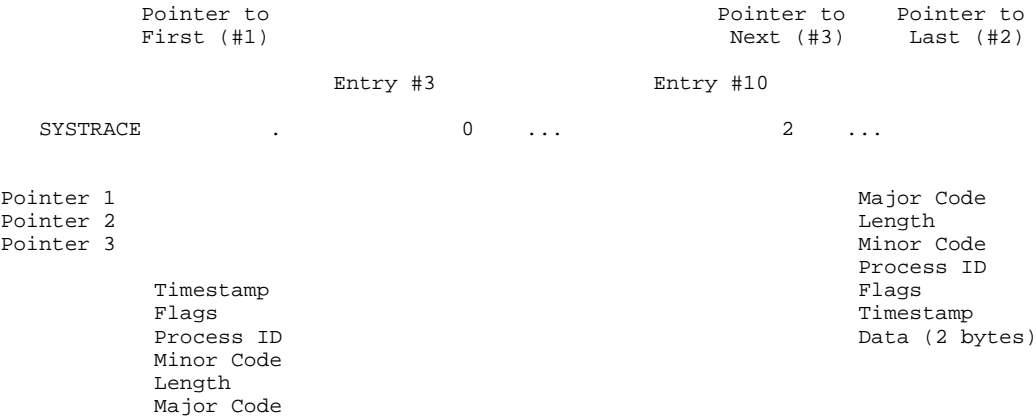
The trace event data contains the information describing each individual trace event. The events traced may be from OS/2 system supplied or other user supplied tracepoints. In either case the data is dependent on each individual tracepoint. Descriptions of the data and formatting instructions for the OS/2 system supplied tracepoints can be found in the [System Tracepoints Reference](#) chapter.

Trace Buffer Structures

Note:

The from OS/2 2.11 fix pack 91 and OS/2 3.0 fix pack 8 the format of the STDA has changed to allow more meaningful time-stamp information. See [New STDA Format](#) at the end of this section for details.

Circular Trace Buffer (STDA)



Field Descriptions: Trace Control Record

Name	#Bytes	Description
ID of Data Area	8	Contains ASCII 'SYSTRACE'
Pointer 1	2	Offset of first byte of the trace buffer
Pointer 2	2	Offset of last byte of the trace buffer
Pointer 3	2	Offset of next available byte in the trace buffer

Field Descriptions: Trace Event Trailer Record (with Timestamp)

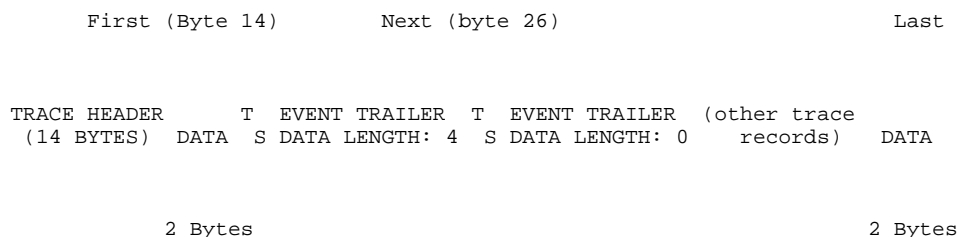
Name	#Bytes	Description
------	--------	-------------

Timestamp	2	Timestamp in seconds and hundredths of seconds (Conditional on bit 1 in the Flags byte)
Flags	1	Trace record flag Bit 0: 0 indicates an internal kernel generated trace record. Bit 1: 0 indicates that a time-stamp is present. Bit 2: 1 signifies that the trace record was generated in protect mode. Bit 3: 0 signifies a static trace record, 1 a dynamic trace record. Bit 4: 1 indicates an incomplete dynamic trace record. Bit 5 - 7: reserved.
PID	2	ID of the process calling the API being traced
Minor Code	2	Minor Event Code
Length	2	Length of data for the traced API
Major Code	1	Major Event Code

### Remarks

The buffer returned by **DosGetSTDA** is a simple circular buffer that is a snapshot of the OS/2 System Trace buffer at the time that the API was called. The actual System Trace buffer is emptied by the call. The buffer contains a header record that has pointers to the FIRST, LAST and NEXT bytes in the buffer. The offsets of the FIRST and LAST bytes are constant and the offset to NEXT is used to indicate the last (most recent) trace record in the buffer. This pointer is logically moved backwards as the buffer is traversed. Since it is possible for a trace record to wrap back to the end of the buffer, it is necessary to look at each part of the data individually (trailer, timestamp and data) to determine whether the length of the data is greater than the distance between NEXT and FIRST. If the length is greater, then the data is continued at the offset to LAST.

For example (see figure below), the buffer has been traversed until the pointer to NEXT is at byte 26. The event trailer record is 8 bytes and the distance from NEXT to FIRST is 12, so the trailer is in contiguous memory. The pointer to NEXT is then set to byte 18. There is a timestamp which is two bytes. Our distance to FIRST is now 4 so the timestamp is contiguous and the pointer to NEXT is reset to 16. This record has 4 bytes of data attached to it. The distance to FIRST at this point is 2, so the data is wrapped: 2 bytes are adjacent to the NEXT, and the other 2 bytes begin at the pointer to LAST.



End of data in the trace buffer is indicated by a trace event trailer that contains a major code field of zero and a length field of zero.

The display format of the OS/2 system supplied tracepoint data is described in the [System Tracepoints Reference](#) chapter. Note that for data using the '%S' (ASCII string) format type, the first byte of the data is reserved, bytes two and three contain the actual length of the string and the string begins at byte 4.

### TRACEFMT Unformatted Trace Buffer

The trace formatter (TRACEFMT) is able to save the unformatted STDA buffer for formatting at a later date. The format of this buffer is as follows:

#### File Header

```
STDA LN      D A T E T I M E      C H E C K   K E Y
0           4                      f                1a
```



STDA

S Y S T R A C E STA END NXT TRACE RECORDS .....

1a                      22    24    26    28

### Remarks

STDA LN                      ULONG length of the STDA read by **DosGetSTDA**. Length is 1 greater than then STDA end offset.

DATETIME                      A DATETIME structure returned by **DosGetDateTime** when this file buffer is created.

CHECK KEY                      The DATETIME field Exclusively ORed with the string constant "TRCFMTBUFF\$"

STDA                          The STDA returned by **DosGetSTDA**.

### Note:

**DosGetSTDA** resets the internal start, end and next offsets after the STDA has been read. This allows trace formatting programs to detect an empty buffer.

For GA OS/2 2.x and 3.x the default start offset is 0x000e.

After fix pack 91 (OS/2 2.11) and fix pack 8 (OS/2 3.0) the the default start offset is 0x001e.

-----

## New STDA Format

From fix pack 8 (OS/2 3.0) and fix pack 91 (OS/2 2.11) the system trace was enhanced to provide improved time-stamp information. Each trace records is time-stamped in hours, minutes, seconds and 1/100 seconds. The trace logging start and stop times are also logged and displayed by the TRACEFMT command.

The spare bytes between the end of the STDA header and first trace record have been reserved for storing trace start and stop times. These are of the following format:

STDA

y y m d h n s c Y Y M D H N S C TRACE RECORDS .....

28                      30                      38

Where:

**yymdhnsc** is the TRACE ON date and time in years, months, days, hours, seconds and 100th seconds.

**YYMDHNSC** is the TRACE OFF date and time in years, months, days, hours, seconds and 100th seconds.

The time-stamp field of a trace record is now 4 bytes and contains in addition hours and minutes. The following diagram compares the different trace records of both old and new formats:

Old format with time-stamp:

Optional Data    s h f Pid Min Len Maj

```

-8      -4      0 1 2 3  5  7  9

```

Old format without time-stamp:

```

Optional Data  f Pid Min Len Maj
-8      -4      0 1  3  5  7

```

New format with time-stamp:

```

Optional Data  H M s h f Pid Min Len Maj
-8      -4      0 1 2 3 4 5  7  9  a

```

where:

Optional Data	is trace data of length specified by the <b>Len</b> field.
H	is time in hours.
M	is time in minutes.
s	is time in seconds.
h	is time in 1/100 seconds.
Pid	is the process id under which the entry was logged. Zero implies interrupt time.
Min	is the minor code.
Len	is the length of the optionally traced data.
Maj	is the major code.

#### Notes:

The format of the buffer used by TRACEFMT has not changed. Thus, as long as a correct header is appended to the extracted STDA then the new TRACEFMT will format the traced data.

The new TRACEFMT will also format trace data of the older format.

STDAs of the new and old formats may be distinguished by the value of the start offset in the STDA header:

For the old format this is 0x000e.

For the new format this is 0x001e.

-----

## DosForceSystemDump (Force a System Stand Alone Dump)

**DosForceSystemDump** allows an application to initiate a stand-alone system dump.

#### Syntax

```

APIRET APIENTRY DosForceSystemDump(ULONG reserved);

32-bit code Example using CSet/2

```

### Parameters

reserved

Reserved doubleword field that is set to 0L.

### Returns.

There is no return from this API.

### Remarks.

There is no return from this API. The system is halted abruptly and a stand-alone dump is initiated. After the stand-alone dump process has completed the system must be re-booted.

No shut down activity is performed when this API is called. File system buffers are not written to disk, cache is not flushed and files are not closed. **Data loss may result.**

**DosForceSystemDump** is equivalent to using the **Ctrl-Alt-Numlock-Numlock** key sequence.

C Language prototype definitions for the **DosForceSystemDump** API may be found under [RAS API Prototypes](#).

To format a system dump see [The Dump Formatter User Guide](#).

For related information see:

- [TRAPDUMP CONFIG.SYS command](#)
- [CREATEDD command](#) in the OS/2 Command Reference

---

## DosDumpProcess (Enable/Disable ProcessDump)

**DosDumpProcess** allows an application:

- to enable or disable dynamically the Process Dump Facility.
- to force a process dump for a given process.

P.The default setting is for Process Dump to be disabled unless overridden by the [DUMPPROCESS CONFIG.SYS command](#).

### Syntax

```
APIRET APIENTRY DosDumpProcess(ULONG Flag, ULONG Drive, PID pid);
```

32-bit code Example using CSet/2

### Parameters

Flag

Doubleword field that may take one of the following values:

- (DDP\_DISABLEPROC\_DUMP 0x00000000L)

disable process dumps

- (DDP\_ENABLEPROC\_DUMP 0x00000001L)

enable process dumps to be taken to a file in the root directory of a drive specified by the *Drive* parameter.

- (DDP\_PERFORMPROC\_DUMP 0x00000002L)

Drive

Doubleword containing the ASCII value of the drive letter to which the PDUMP.nnn dump files will be written when DDP\_ENABLEPROC\_DUMP is specified. For DDP\_DISABLEPROC\_DUMP this parameter is ignored.

pid

Doubleword containing the process Id of the process to be dumped.

This option is valid only with DDP\_PERFORMPROC\_DUMP. If zero is specified for **Pid** then the current process is dumped.

#### Returns.

Return Code.

**DosDumpProcess** returns the following values:

0	NO_ERROR
87	ERROR_INVALID_PARAMETER
303	ERROR_INVALID_PROCID

#### Remarks.

When Process dump is enabled a dump file is written whenever a ring 3 process traps. The file takes a name **PDUMP.nnn** where **nnn** is incremented sequentially (starting from 000) for each successive dump.

The directory to which PDUMP.nnn will be written is always the root directory of *Drive*.

C Language prototype definitions for the **DosDumpProcess** may be found under [RAS API Prototypes](#).

The content of a Process Dump comprises register information at time of trap, system control blocks (TCB, TSD, PTDA, MTE, SMTE, OTE, VMAR, VMOB, LTD) that describe the state of the process at the time of error, ring 0 and ring 3 stack data for the trapping process.

See the [Process Dump Formatter](#) section of the Dump Formatter User Guide for information on formatting Process Dumps.

#### **Note:**

DDP\_PERFORMPROC\_DUMP is not available in some early releases of OS/2 V2.11

---

## DosSuppressPopUps (Suppress Trap Exception Pop-Up Messages)

**DosSuppressPopUps** allows an application to enable or disable dynamically Trap Exception pop-up suppression and to specify the drive where the pop-up suppression log will be recorded.

The default setting is for disabled pop-up suppression unless overridden by the [SUPPRESSPOPUPS CONFIG.SYS command](#).

#### Syntax

```
APIRET APIENTRY DosSuppressPopUps(ULONG Flag, ULONG Drive);  
  
32-bit code Example using CSet/2
```

#### Parameters

Flag

Doubleword field that may take one of the following values:

- (SPU\_DISABLESUPPRESSION 0x00000000L)

Disable pop-up suppression

- (SPU\_ENABLESUPPRESSION 0x00000001L)

Enable pop-up suppression and pop-up logging to file POPUPLOG.OS2 on drive specified by the *Drive* parameter.

Drive

Doubleword containing the ASCII value of the drive letter to which the POPUPLOG.OS2 log file will be written when SPU\_ENABLESUPPRESSION is specified. With SPU\_DISABLESUPPRESSION, **Drive** is ignored.

#### Returns.

Return Code.

**DosSuppressPopups** returns the following values:

0	NO_ERROR
87	ERROR_INVALID_PARAMETER

#### Remarks.

The directory to which POPUPLOG.OS2 will be written is always the root directory of *Drive*.

A prototype definition of **DosSuppressPopUps** may be found under [RAS API Prototypes](#).

See also **DosError** API in the Control Program Programming Reference.

---

## DosQueryRASInfo (Query RAS Information)

**DosQueryRASInfo** returns information about active trace event recording and System Logging facility from the [Global Information Segment \(InfoSegGDT\)](#). dump.

#### Syntax

```
APIRET  APIENTRY DosQueryRASInfo(ULONG Index, PPVOID Addr);  
  
32-bit code Example using CSet/2
```

#### Parameters

Index

Doubleword field that may take one of the following values:

- (SPU\_SIS\_MEC\_TABLE 0x00000001L)

Return the address of the table of actively traced major event codes in the InfoSegGDT. The table is 32 bytes long, each bit represents each major event code from 0 to 255.

- (SIS\_SYS\_LOG 0x00000002L)

Return the address of the SYSLOG status word from the **InfoSegGDT**. The status may contain a combination of:

- (LF\_LOGENABLE 0x0001) Logging enabled
- (LF\_LOGAVAILABLE 0x0002) Logging available

#### Returns.

Return Code

**DosQueryRASInfo** returns the following values:

0	NO_ERROR
5	ERROR_ACCESS_DENIED
87	

ERROR\_INVALID\_PARAMETER

#### Remarks.

For related information see:

- [Logging Facility](#)
- [The OS/2 Trace Facility](#)

-----

## 16 Bit Error Logging API's for IBM OS/2 Version 2.1

This section describes the "Logging Facility for OS/2 2.1". This comprises a set of 3 APIs, the logging daemon (LOG.SYS) and the log formatter (SYSLOG).

Both the Logging Daemon and Log Formatter are described in the OS/2 Command Reference - see [LOG.SYS](#) under DEVICE statement of CONFIG.SYS and the [SYSLOG](#) command.

#### **Note:**

C Language prototype definitions for the Error Logging APIs may be found under [RAS API Prototypes](#).

The following topics are described in this section:

- [Static vs Dynamic Error Log Record I.D. Registration](#)
- [DosLogRegister](#) API
- [DosLogEntry](#) API
- [DosLogRead](#) API
- [Error Log Entry Formatting DLL Routines](#)

-----

## Dynamic vs. Static Error Log Record I.D. Registration

OS/2 2.0 users of the **DosLogEntry** API will not need to use the **DosLogRegister** API. The **DosLogRegister** API is only maintained on OS/2 2.0 to support existing OS/2 1.3 programs that did need to use the API.

The OS/2 2.0 version of the **DosLogRegister** API will always return a "default" Error Log record I.D.. It will accept a format template string as an input, but it will do nothing with the string since format template strings will not be saved within the OS/2 2.0 version of the Error Log file.

The OS/2 2.0 version of the **DosLogEntry** API will behave similarly to the OS/2 1.3 version of the API. Since the OS/2 2.0 version of the system Error Logging facility no longer supports the saving of format template strings within the Error Log file, it is necessary to provide a method by which **DosLogEntry** callers can associate their Error Log entry with a formatting (.DLL) routine. The OS/2 2.0 version of the **DosLogEntry** API will make a special interpretation of the Originator Name field within the packet header. It will be assumed that this name field (if not NULL) contains the name of a Error Log formatting .DLL module.

-----

## DosLogRegister

There are two major differences between the OS/2 2.0 version of **DosLogRegister** and the 1.3 version of the API:

- **DosLogRegister** no longer supports dynamic registration of Error Log record I.D.'s. Instead, the API always returns a single

"default" value.

- **DosLogRegister** no longer supports entry format template registration. While the API still accepts a format template as part of its input data packet, the format template will not be acted upon in any way.

**DosLogRegister** continues to support the existing alert notification registration function.

The description of the OS/2 2.0 version of the **DosLogRegister** API follows:

#### Syntax

```
APIRET16 APIENTRY16 DosLogRegister((PUSHORT) LogHandle,  
                                   (PVOID) LogRegList,  
                                   (PUSHORT) RequestID)
```

32-bit code Example using CSet/2

#### Parameters

##### **LogHandle**

The address of the word in which the system will return the handle of a named pipe that will be transparently used in subsequent **DosLogRead** calls.

##### **LogRegList**

The address of the log registry buffer.

##### **RequestID**

The address of the word that the system will fill in with a "default" Error Log record I.D. (if the 'Error Log record I.D.' field in the log registry buffer is set by the caller to -1)

#### Returns

Return code

**DosLogRegister** returns the following values

0

Success

non-zero

Failure.

Possible reasons for failure are:

Facility unavailable

Record I.D. in use

Registration failed (general failure)

Invalid I.D.

Too many open files

Too many semaphores

Semaphore not found

User semaphore limit reached

Request timed out without satisfaction

Error Log buffer temporarily full

#### Remarks

Log Registry Buffer format description:

length of the registration data

reserved	2
Error Log record I.D.	2
offset to the format template layout field	2
semaphore name string	variable length
format template layout	variable length

Where:

**'length of the registration data'**

is the total number of bytes in the current Log Registry Buffer (this length includes the two byte length field itself)

**'reserved'**

is a two byte reserved field

**'Error Log record I.D.'**

contains the Error Log record I.D. that caller wishes to be registered for. If the field is set to 0xFFFF (-1), then a "default" record I.D. is returned in the word pointed to by the 'RequestID' parameter. This field can be used to specify an alert notification record I.D. (that is, the caller wishes to be alerted whenever an Error Log Entry containing this record I.D. is logged).

**'offset to the format template layout field'**

is the offset within the Log Registry Buffer to the start of the format template layout area.

**'semaphore name string'**

is the name of a system semaphore, created with the nonexclusive option, that will be used to alert the caller's process when an Error Log entry containing the specified 'Error Log record I.D.' is logged. The name string is an ASCII string.

**'format template layout'**

is an area within the Log Registry Buffer that contains the formatting structure information that is placed within the 1.3 Error Log file. This area is not used in the OS/2 2.0 version of the **DosLogRegister** call. However, the 'length of the registration data' field should reflect the size of this area.

In order to resolve successfully **DosLogRegister** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSLOGREGISTER=DOSCALL1.195
```

## DosLogEntry

There are two major differences between the OS/2 2.0 version of **DosLogEntry** and the 1.3 version of the API:

- Since the **DosLogRegister** API will only return a "default" Error Log record I.D. to its caller, the **DosLogEntry** caller must override this "default" record with the appropriately statically allocated record I.D. if the caller wishes to see the "correct" record I.D. in the Error Log record.
- Since there is no explicit "Error Log record formatting DLL module name" field in the **DosLogEntry** log data packet, the API will attempt to interpret the 'Originator Name' field in the packet's header portion as a formatting DLL module name.

The description of the OS/2 2.0 version of the **DosLogEntry** API follows:

**Syntax**

```
APIRET16 APIENTRY16 DosLogEntry((USHORT) Function,
```



(PVOID) LogData)

32-bit code Example using CSet/2

### Parameters

#### Function

specifies the type of log entry:

0H	Reserved
1H	Error Logging
2H-FFFFH	Reserved

#### LogData

is the address of the log data buffer that contains one or more variable length log packets.

### Returns

Return Code.

**DosLogEntry** returns the following values:

0	Success
non-zero	Failure
	Possible reasons for failure:
	Invalid function
	Facility unavailable
	Facility suspended
	Error Log buffer temporarily full

### Remarks

Error Log Data Buffer format description:

Multiple log packets can be included within a single log data buffer. In the following diagram, the size of each field is indicated in bytes:

# of log packets (within the buffer)	2	
length of the current log packet	2	<
Error Log record I.D.	2	
time of logging	4	multiple log packets within a single log data buffer
date of logging	4	
originator name	8	
qualifier name	4	
Error Log entry data	<= 1024 <	

Where:

#### '# of log packets'

is the number of separate packets contained within the user's buffer

#### 'length of the current log packet'

is the number of bytes in the current log packet within the user's log data buffer (this length includes the length of all the log packet control fields and the size of the Error Log entry data).

**'Error Log record I.D.'**

is the record I.D. for the current Error Log entry (I.D. registration will be statically registered by the OS/2 development organization). The caller may pass in the "default" Error Log record I.D. that is returned by the **DosLogRegister** API.

**'time of logging'**

is filled in by the system Error Logging facility )

**'date of logging'**

is filled in by the system Error Logging facility

**'originator name'**

is a primary name field that is provided by the caller

**'qualifier name'**

is a secondary name field that is provided by the caller

**'Error'**

Log entry data' is an optional variable length set of data that can be supplied by the caller (the format of the data is under the control of the caller).

In order to successfully resolve **DosLogEntry** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSLOGENTRY=DOSCALL1.193
```

-----

## DosLogRead

The description of the OS/2 2.0 version of the **DosLogRead** API follows:

```
APIRET16 APIENTRY16 DosLogRead((USHORT) LogHandle,
                                (USHORT) Length,
                                (PVOID) LogBuffer,
                                (PUSHORT) ReadSize)
```

32-bit code Example using CSet/2

### Parameters

**LogHandle**

is the named pipe handle returned by **DosLogRegister()**

**Length**

is the length (in words) of the caller's log buffer

**LogBuffer**

is the address of the caller's buffer, into which the system Error Logging facility will place a single Error Log entry packet (formatted in the manner of the 16 bit **DosLogEntry** API).

**ReadSize**

is the address of a word, into which the system Error Logging facility will place the number of bytes that it wrote into the caller's log buffer. If a zero is returned here, then there was no Error Log packet to return.

### Returns

Return code

**DosLogRead** returns the following values:

0 indicating Success.

**non-zero** indicating error

Possible reasons for failure:

Invalid log handle

Facility unavailable

Buffer too small

In order to resolve successfully **DosLogRead** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
DOSLOGREAD=DOSCALL1.196
```

**DosLogRead** returns Error Log entries that are formatted in the manner of the 16 bit **DosLogEntry** API.

## Error Log Entry Formatting DLL Routines

Each Error Log record within an Error Log file can contain the name of a formatting DLL module. A formatting DLL module is invoked by the SYSLOG utility when SYSLOG encounters an Error Log record that contains the name of the DLL module.

Each formatting module contains a single formatting routine that can be identified by an ordinal value of 1. The formatting routine can be designed to handle a single type of Error Log entry or to handle multiple types of Error Log entries. When SYSLOG passes control to a formatting routine, it passes the entire Error Log record (both header portion and data portion) to the formatting routine. The formatting routine has the complete flexibility to format an Error Log entry as it deems appropriate.

SYSLOG uses the **DosLoadModule** API to create a run-time link to the specified formatting DLL module. It uses the **DosFreeModule** API to free the DLL module after it receives its response from the formatting routine.

There are no specific rules that govern the naming of a formatting DLL module. However, since it is desirable to reduce the possibility of "colliding" with another DLL module of the same name, it is suggested that a formatting DLL module be labelled with a name that adheres to the following standard form:

ELGxxxxx.DLL (where "xxxxxx" corresponds to the Error Log record I.D. (in decimal) of any one of the types of records that the formatting routine is designed to handle)

for example, "ELG00127.DLL" is a standardized name for a formatting DDL module that recognizes (among other things) Log records with I.D. of 127 (decimal)

This standard naming convention is suggested because it is assumed that the Error Log records of any one I.D. will only be recognized by a single formatting routine. Therefore the use of the "xxxxx" suffix (based on record I.D.) should assure uniqueness for the formatting module name.

The static Error Log record I.D. registration mechanism that is enforced by the OS/2 development organization will attempt to keep a list not only of the Error Log record I.D.'s in use, but also the names of the formatting DLL modules that correspond to each record I.D.. This will also help to reduce the possibility of formatting DLL module names "colliding".

In addition to its single formatting routine, each formatting DLL module must contain a global variable named "ELOG\_FORMAT". For OS/2 2.0, this exported global variable must be set to a value of 1. When SYSLOG loads a prospective formatting DLL module it attempts to

access this global variable and check whether it has the expected value of 1. If the global variable check fails, then SYSLOG can conclude that it has accidentally loaded another DLL module with the same name as the formatting module that is mentioned in the Error Log entry. This check is intended as a form of protective validation for SYSLOG. The variable will in future releases be used as a revision level for the SYSLOG/formatting DLL module interface specification.

When a user constructs a Error Log entry formatting DLL module, care should be taken not to export the names of its constituent formatting routine (though the required ELOG\_FORMAT global variable must be exported). Not exporting the module name will save storage space within the OS/2 kernel. The SYSLOG utility will be written to use the "ordinal" version of the **DosGetProcAddr** API.

Error Log record formatting DLL routines must be written as 32 bit procedures. A typical Error Log record formatting DLL routine will have to accept the parameters:

```
ULONG ELGxxxxx((PVOID) Log_Record, (PVOID) String_Buffer,  
               (ULONG) Buffer_Length, (PULONG) String_Length)
```

### Parameters

#### **Log\_Record**

a linear pointer to an Error Log record that is being passed from SYSLOG to the formatting routine. The Error Log record adheres to the format that is described in the section that follows entitled "Error Log File Entry Format", except that the linear pointer points to the "TOT\_LENGTH" field (since the "PREV\_PTR" and "PREV\_SIZE" fields are of no interest to a formatting routine).

#### **String\_Buffer**

is a linear pointer to a buffer provided by SYSLOG so that the formatting routine can return a series of ASCIIZ strings to SYSLOG. Each ASCIIZ string should correspond to a line of formatted display. Each ASCIIZ string should be limited to a maximum of 80 characters. SYSLOG will paint each string "line" within its client window. The strings should not contain NEWLINE characters. SYSLOG will automatically format the header portion of the Error Log entry. The formatted output prepared by this routine will follow the formatted header display.

#### **Buffer\_Length**

is a 32 bit integer that contains the maximum size of the 'String\_Buffer'.

#### **String\_Length**

is a pointer to a 32 bit integer that is set by the formatting routine to the total length of the ASCIIZ strings that have been placed in 'String\_Buffer'.

### Returns

**ELGxxxxx** returns the following:

**0**

indicating success.

**-1**

indicates insufficient space in 'String\_Buffer' positive values indicate formatting routine errors.

If a formatting DLL routine returns a positive error code to SYSLOG, SYSLOG will format the header portion of the Error Log record in the standard manner, display the returned formatting routine error code (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If an Error Log record fails to point to a formatting DLL module, or if the formatting DLL module cannot be successfully loaded and validated, then SYSLOG will format the header portion of the Error Log record in the standard manner, display a message that a formatting routine was not specified or could not be successfully invoked (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If there is insufficient space in the 'String\_Buffer', then the formatting routine will return a -1 status code, and will place the required length of the formatted display string in the caller's output length variable. SYSLOG can react to this error by recalling the formatting routine with a larger 'String\_Buffer'.

SYSLOG will contain logic to format the standard SNA Generic Alert entry (that is, Error Log record I.D. of 2). This is necessary since most of the existing Error Log calls are used to pass generic alerts (and the existing calls can not pass in formatting DLL routine names). This design choice does not prevent future Error Log callers to specify a record I.D. of 2 and also to pass in the name of a formatting DLL routine that knows how to specially format that Generic Alert entry.

---

## 32-Bit Error Logging API's for IBM OS/2 Version 2.1 and 3.0

This section describes the "Logging Facility for OS/2 2.1 and 3.0". This comprises a set of 4 APIs, a DevHlp function, the logging daemon (LOGDAEM.EXE), logging device driver (LOG.SYS) and the log formatter (SYSLOG).

The Logging Daemon, Device Driver and Log Formatter are described in the OS/2 Command Reference - see [LOG.SYS](#) under DEVICE statement of CONFIG.SYS and the [SYSLOG](#) command.

**Note:**

C Language prototype definitions for the Error Logging APIs may be found under [RAS API Prototypes](#).

The following topics are described in this section:

- [LogOpen](#) API
- [LogClose](#) API
- [LogAddEntries](#) API
- [32-bit Error Log Entry Formatting DLL Routines](#)
- [DevHlp\\_LogEntry](#) Device Driver interface

The set of four 32-bit logging APIs provide equivalent functionality to the three 16-bit logging APIs discussed in the previous section. They may be used as a complete replacement to the 16-bit set.

-----

## LogOpen

**LogOpen** is a 32-bit system Error Logging facility high level API. It is used to open a connection to the facility (through the System Logging Service device driver).

The description of the **LogOpen** API call follows:

**Syntax**

```
APIRET APIENTRY LogOpen(PHFILE phf);
```

**Parameters**

**phf**

points to a file handle holder that on return will hold an open file handle

**Returns**

Return code.

**LogOpen** returns the following values:

0

Success.

non-zero

Facility not available.

**Remarks**

The file handle that is returned by the **LogOpen** API is required in all subsequent high level system Error Logging facility API calls.

In order to resolve successfully **LogOpen** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
LogOpen=DOSCALL1.430
```

---

## LogClose

**LogClose** is a 32-bit system Error Logging facility high level API. It is used to close a connection to the facility.

The description of the **LogClose** API call follows:

### Syntax

```
APIRET APIENTRY LogClose(HFILE hf);
```

### Parameters

hf  
is the file handle returned by **LogOpen()**

### Returns

Return code.

**LogClose** returns the following values:

0  
Success.

non-zero  
Failure. Possible reason: facility not open.

### Remarks

In order to resolve successfully **LogClose** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
LogClose=DOSCALL1.431
```

---

## LogAddEntries

**LogAddEntries** is a 32-bit system Error Logging facility high level API. It is used to allow application processes to add Error Log entries to the internal Error Log buffer that is maintained by the System Logging Service device driver.

The description of the **LogAddEntries** API call follows:

### Syntax

```
APIRET APIENTRY LogAddEntries(HFILE hf, ULONG service,
                              PVOID log_data_address);
```

### Parameters

hf  
is the file handle returned by **LogOpen()**

service  
specifies the class of logging facility:

0x0	Reserved
0x1	Error Logging
0x2 - 0xffff	Reserved

log\_data\_address  
is the address of a buffer that contains a variable length Error Log entry. The first word of the buffer contains the number of packets in the Error Log entry

### Returns

Return code.

**LogAddEntries** return the following values:

0	Success
non-zero	Failure

Possible reasons for failure are:

- Invalid log type
- Facility unavailable
- Facility suspended
- Facility not open
- Error Log buffer temporarily full

### Remarks

Error Log Entry Buffer format description:

Multiple Error Log packets can be included within a single Error Log entry buffer. If multiple packets are included within a single buffer, each individual packet should be aligned on a double word boundary. In the following diagram, the size of each field is indicated in bytes:

packet revision number	2	
# of Error Log entry packets	2	
length of this Error Log entry packet	2	<
Error Log record I.D.	2	
status flags	4	
qualifier name	4	multiple
reserved	4	Error
time of logging	4	Log entry
		packets
		within a

date of logging	4	single Error Log Entry Buffer
originator name	8 or 256	
process name (optional)	0 or 260	
formatting DLL module name (optional)	12	
Error Log entry data	<= 3400 <	

Where

'packet revision number'

is an integer value that can be used to distinguish error logging packets that are intended to be handled by different revisions of the **LogAddEntries** API. For the initial version of the API, this field should be set to a value of 1. This field is included in the packet to support future backward compatibility.

'# of Error Log entry packets'

is the number of separate packets contained within the user's buffer.

'length of this Error Log entry packet'

is the number of bytes in the current Error Log entry packet within the user's Error Log Entry Buffer (this length includes the length of all the Error Log entry packet control fields and the size of the Error Log entry text). To support efficient logging execution, this length should be a multiple of 4 bytes (i.e. if necessary the user should pad the Error Log entry packet).

'Error Log record I.D.'

is the record I.D. for the current Error Log entry (I.D. registration will be statically registered by the OS/2 development organization).

'status flags'

is a two byte flag holder that contains three single bit flags:

(BIT 0) is used to indicate whether the current Error Log entry packet contains space in which the Error Logging facility can place a long process name ("on" indicates YES, "off" indicates NO);

(BIT 1) is used to indicate whether the current Error Log entry packet contains an 8 byte originator name or a 256 byte originator name ("on" indicates a 256 byte originator name, "off" indicates an 8 byte originator name);

(BIT 2) is used to indicate that the caller has placed time and date values in the Error Log entry packet and does not wish to have those values modified during the logging process ("on" indicates that the Error Log entry packet already contains time and date values, "off" indicates the packet does not already contain time and date values);

All the other 29 bits in 'status flags' are considered reserved at this time and will be zeroed by the **LogAddEntries** API.

'qualifier name'

is a secondary name field that is provided by the caller

'reserved'

is a four byte reserved field

'time of logging'

is filled in by the system Error Logging facility (unless BIT 2 of the 'status flags' field is "on", indicating that the caller has preset a time value).

'date of logging'

is filled in by the system Error Logging facility (unless BIT 2 of the 'status flags' field is "on", indicating that the caller has preset a date value);

'originator name'

is a primary name field that is provided by the caller.

'process name'

is an optional long process name field that will be filled in by the Error Logging facility if the field is provided by the caller in the Error Log entry packet.

'formatting DLL module name'

is the optional name of a DLL module that houses a formatting routine that recognizes this type of Error Log entry



and can format it for display by the SYSLOG utility. The name is specified as an ASCIIZ string that can be up to eight characters in length. If no module name is specified in this field, then SYSLOG will display the data portion of the Error Log entry as a hexadecimal dump.

'Error Log entry data'

is an optional variable length set of data that can be supplied by the caller (the format of the string is under the control of the caller).

The format and function of the **LogAddEntries** API call is very similar to that of the 16-bit **DosLogEntry** call. There are several functional differences from the **DosLogEntry** call:

- The user-supplied error log entry Record I.D. will now be a statically allocated value rather than a dynamically allocated value.
- The maximum size of the originator name field in the caller's packet has been increased from 8 bytes to 256 bytes. The caller can specify whether the packet contains an 8 byte originator name field or a 256 byte originator name field.
- The maximum size of the variable length data portion within the caller's packet has been increased from 1024 bytes to 3400 bytes
- The order of the fields within the Error Log entry has been slightly rearranged to support the creation of smaller internal control messages.

In order to resolve successfully **LogAddEntries** function calls in your program, the following lines must be added to the Linker Definition (DEF) file:

```
IMPORTS
LogAddEntries=DOSCALL1.432
```

---

## 32-Bit Error Log Entry Formatting DLL Routines

Each Error Log record within an Error Log file can contain the name of a formatting DLL module. A formatting DLL module is invoked by the SYSLOG utility when SYSLOG encounters an Error Log record that contains the name of the DLL module.

Each formatting module contains a single formatting routine that can be identified by an ordinal value of 1. The formatting routine can be designed to handle a single type of Error Log entry or to handle multiple types of Error Log entries. When SYSLOG passes control to a formatting routine, it passes the entire Error Log record (both header portion and data portion) to the formatting routine. The formatting routine has the complete flexibility to format an Error Log entry as it deems appropriate.

SYSLOG uses the **DosLoadModule** API to create a run-time link to the specified formatting DLL module. It also uses the **DosFreeModule** API to free the DLL module after it receives its response from the formatting routine.

There are no specific rules that govern the naming of a formatting DLL module. However, since it is desirable to reduce the possibility of "colliding" with another DLL module of the same name, it is suggested that a formatting DLL module be labeled with a name that adheres to the following standard form:

```
ELGxxxxx.DLL      (where "xxxxx" corresponds to the Error
                   Log record I.D. (in decimal) of any one
                   of the types of records that the formatting
                   routine is designed to handle)
```

```
e.g. "ELG00127.DLL" is a standardized name for a formatting
    DDL module that recognizes (among other things) Error
    Log records with I.D. of 127 (decimal)
```

This standard naming convention is suggested because it is assumed that the Error Log records of any one I.D. will only be recognized by a single formatting routine. Therefore the use of the "xxxxx" suffix (based on record I.D.) should assure uniqueness for the formatting module name.

The static Error Log record I.D. registration mechanism that is enforced by the OS/2 RAS development group will attempt to keep a list not

only of the Error Log record I.D.'s in use, but also the names of the formatting DLL modules that correspond to each record I.D.. This may also help to reduce the possibility of formatting DLL module names "colliding".

In addition to its single formatting routine, each formatting DLL module must contain a global variable named "ELOG\_FORMAT". This exported global variable must be set to a value of 1. When SYSLOG loads a prospective formatting DLL module it will attempt to access this global variable and check whether it has the expected value of 1. If the global variable check fails, then SYSLOG can conclude that it has accidentally loaded another DLL module with the same name as the formatting module that is mentioned in the Error Log entry. This check is intended as a form of protective validation for SYSLOG. The variable may in future releases be used as a sort of revision level for the SYSLOG/formatting DLL module interface specification. That is why it will initially be forced to a value of 1.

When a user constructs a Error Log entry formatting DLL module, care should be taken not to export the names of its constituent formatting routine (though the required ELOG\_FORMAT global variable must be exported). Not exporting the module name will save storage space within the OS/2 kernel.

Error Log record formatting DLL routines must be written as 32-bit procedures. A typical Error Log record formatting DLL routine will have to accept the parameters:

```
APIRET APIENTRY ELGxxxxx(PVOID Log_Record,  
                          PVOID String_Buffer,  
                          ULONG Buffer_Length,  
                          PULONG String_Length);
```

### **Parameters**

Log\_Record

is a linear pointer to an Error Log record that is being passed from SYSLOG to the formatting routine. The Error Log record adheres to the format that is described in the section that follows entitled "Error Log File Entry Format", except that the linear pointer points to the "TOT\_LENGTH" field (since the "PREV\_PTR" and "PREV\_SIZE" fields are of no interest to a formatting routine).

String\_Buffer

is a linear pointer to a buffer provided by SYSLOG so that the formatting routine can return a series of ASCIIZ strings to SYSLOG. Each ASCIIZ string should correspond to a line of formatted display. Each ASCIIZ string should be limited to a maximum of 80 characters. SYSLOG will paint each string "line" within its client window. The strings should not contain NEWLINE characters. SYSLOG will automatically format the header portion of the Error Log entry. The formatted output prepared by this routine will follow the formatted header display.

Buffer\_Length

is a 32 bit integer that contains the maximum size of the 'String\_Buffer'.

String\_Length

is a pointer to a 32 bit integer that is set by the formatting routine to the total length of the ASCIIZ strings that have been placed in 'String\_Buffer'.

### **Returns**

ELGxxxxx returns the following:

0

indicating success.

-1

indicates insufficient space in 'String\_Buffer' positive values indicate formatting routine errors.

### **Remarks**

If a formatting DLL routine returns a positive error code to SYSLOG, SYSLOG will format the header portion of the Error Log record in the standard manner, display the returned formatting routine error code (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If an Error Log record fails to point to a formatting DLL module, or if the formatting DLL module cannot be successfully loaded and validated, then SYSLOG will format the header portion of the Error Log record in the standard manner, display a message that a formatting routine was not specified or could not be successfully invoked (as a line within the formatted display), and then format the data portion of the Error Log record as a hexadecimal dump.

If there is insufficient space in the 'String\_Buffer', then the formatting routine will return a -1 status code, and will place the required length of the formatted display string in the caller's output length variable. SYSLOG can react to this error by recalling the formatting routine with a larger 'String\_Buffer'.

SYSLOG contains logic to format the standard SNA Generic Alert entry (i.e. Error Log record I.D. of 2). This is necessary since most of the existing Error Log calls are used to pass generic alerts (and the existing calls can not pass in formatting DLL routine names). This design choice does not prevent future Error Log callers to specify a record I.D. of 2 and also to pass in the name of a formatting DLL routine that knows how to specially format that Generic Alert entry.

## DevHlp\_LogEntry Device Driver Interface

DevHlp\_LogEntry provides a device driver interface to the logging facility.

The description of the LogEntry DevHlp function follows:

```
Calling sequence -  LES    BX,log_data_address
                   MOV    CX,service
                   MOV    DL,DevHlp_LogEntry /* LogEntry function
                                           code 0x3b */
                   CALL   [Device_Help]
```

### Parameters

log_data_address	is the address of a buffer that contains a variable length Error Log entry. (See the section on the LogAddEntries high level API for further details.)	
service	the class of logging facility:	
	0x0	Reserved
	0x1	"Old-Style" Error Logging call ("old" 16-bit (DosLogEntry-style) data packet provided).
	0x2 - 0x2f	Reserved for future use.
	0x80 - 0x8f	Reserved for internal use by the System Logging Service device driver.
	0x90	"New_Style" Error Logging call ("new" 32-bit (LogAddEntries-style) data packet provided).
	0x91 - 0xffff	Reserved for future use.

### Returns

Return code in AX:

0	Success
non-zero	Failure.
	Possible errors:
	Invalid log type
	Facility unavailable
	Facility suspended

### Remarks

When CX is set to 80H, DS:SI is set to point to the device driver header block of the System Logging Service device driver.

# RAS API Prototypes

The following is a sample C language header file that contains sample prototype definitions for the RAS APIs.

```
/* definitions for DosDumpProcess */
#define DDP_DISABLEPROC_DUMP 0x00000000L /* disable process dumps */
#define DDP_ENABLEPROC_DUMP 0x00000001L /* enable process dumps */
#define DDP_PERFORMPROC_DUMP 0x00000002L /* perform process dump */

/* definitions for DosSuppressPopUps */
#define SPU_DISABLESUPPRESSION 0x00000000L /* disable popup suppression */
#define SPU_ENABLESUPPRESSION 0x00000001L /* enable popup suppression */

/* definitions for DosQueryRASInfo Index */
#define SIS_MMIOADDR 0
#define SIS_MEC_TABLE 1
#define SIS_SYS_LOG 2
#define LF_LOGENABLE 0x0001 /* Logging enabled */
#define LF_LOGAVAILABLE 0x0002 /* Logging available */

APIRET APIENTRY DosQueryRASInfo(ULONG Index, PPVOID Addr);

APIRET APIENTRY DosForceSystemDump(ULONG reserved);

APIRET APIENTRY DosDumpProcess(ULONG Flag, ULONG Drive, PID Pid);

APIRET APIENTRY DosSuppressPopUps(ULONG Flag, ULONG Drive);

APIRET16 APIENTRY16 DosSysTrace(USHORT Majorcode, USHORT Length,
                                USHORT Minorcode, PCHAR pData);

APIRET16 APIENTRY16 DosGetSTDA(SEL, SHORT, SHORT );

/* 32-bit Logging Facility Function Prototypes */

/*-----*/
/* Logging Defines */
/*-----*/
#define ERRLOG_SERVICE 1L
#define ERRLOG_VERSION 1

/*-----*/
/* LogRecord status bits */
/*-----*/
#define LF_BIT_PROCNAME 0x0001L
#define LF_BIT_ORIGIN_256 0x0002L
#define LF_BIT_DATETIME 0x0004L
#define LF_BIT_SUSPEND 0x0008L
#define LF_BIT_RESUME 0x0010L
#define LF_BIT_REDIRECT 0x0020L
#define LF_BIT_GETSTATUS 0x0040L
#define LF_BIT_REGISTER 0x0080L
#define LF_BIT_REMOTE_FAIL 0x0100L

/*-----*/
/* Log Entry Record Header for 2.X */
/* This is format used by 2.0 device */
/* drivers and callers of LogAddEntries */
/*-----*/
typedef struct LogRecord
{
    USHORT len ; /* this record length(includes len field)*/
    USHORT rec_id ; /* record id */
    ULONG status ; /* record status bits(see LF_BIT_) */
    UCHAR qualifier[4] ; /* qualifier tag */
    ULONG reserved ;
    ULONG time ; /* hours minutes seconds hundreds */
    ULONG date ; /* day month (USHORT)year */
    UCHAR data[1] ; /* begin of variable data that includes: */
                    /* Originator(256 bytes if LF_BIT_ORIGIN_256)*/
                    /* else 8 bytes long */
                    /* Processname(260 bytes) only if status */
                    /* LF_BIT_PROCNAME set */
}
```

```

/* FormatDLLName(12 bytes) */
/* Variable data */
} LOGRECORD ;
typedef LOGRECORD far *PLOGREC ;

/*-----*/
/* Format of buffer sent to LogAddEntries */
/*-----*/
typedef struct LogEntryRec
{
    USHORT    version ;                /* this version is 1 */
    USHORT    count ;                 /* number of log records in this buffer */
    LOGRECORD logrec ;                /* repeated count times */
} LOGENTRYREC ;
typedef LOGENTRYREC far *PLOGENTRYREC ;

/*-----*/
/* Logging facility Function prototypes */
/*-----*/
APIRET APIENTRY LogOpen( PHFILE phf );

APIRET APIENTRY LogClose( HFILE hf );

APIRET APIENTRY LogAddEntries( HFILE hf, ULONG ulService, PVOID pLogEntries );

/* 16-bit Logging Facility Function Prototypes */

APIRET16 APIENTRY16 DosLogRegister(PUSHORT LogHandle,
                                   PVOID LogRegList,
                                   PUSHORT RequestID);

APIRET16 APIENTRY16 DosLogEntry(USHORT Function,
                                PVOID LogData);

APIRET16 APIENTRY16 DosLogRead(USHORT LogHandle,
                               USHORT Length,
                               PVOID LogBuffer,
                               PUSHORT ReadSize);

```

## OS/2 System Control Block Reference

This chapter contains details of some of the more important system control blocks used in debugging.

Where major differences in format exist between **ALLSTRICT** and **RETAIL**, and between versions of OS/2 then each version of the control block is given. Otherwise only OS/2 Warp V3.0 ALLSTRICT kernel versions of the control blocks are given and may be assumed to be applicable to also **RETAIL** and earlier versions of OS/2.

### Warning:

The information given in this is for debugging purposes only. The layout of the control blocks may change from one release of OS/2 to the next. They are not to be considered a programming interface.

The following System Components are included in this chapter and an overview is provided in the next section: [Overview of Kernel Components and Interfaces](#).

### Miscellaneous System Control Blocks

This section describes system structures that are common to all components. These include: SAS and RMP.

### Semaphore Management

This section describes the control blocks used for RamSem, FSRamSem Ksem, SysSem, PM/GRE, 32-bit, and MuxWait Semaphores.

### Memory Management

This section describes the following control blocks used by Memory Management:

VMAL, VMOB, VMAR, VMCO, VMAT, VMAH, VMKH, PAI, PGDATA, PF and VP.

#### Scheduler/Dispatcher

This section describes the following control blocks used by Thread and Process Management:

PTDA, TCB, TSD, ljmp, GISEG, LISEG, PIB, TIB, EXENT and Exception Handler structures.

#### System Loader

This section describes the following control blocks used by the System Loader component:

MTE, SMTE, OTE, STE

#### File System

This section describes the following control blocks used by the File System component:

SFT, MFT, FSC, RLR, VPB, DBP, CDS, BUF, Named and Anonymous Pipes.

#### I/O and Device Driver

This section describes the structures that relate to low level I/O. These include: Request Packets, BIOS Parameters Blocks and Device Driver Headers, Virtual Device Driver Entry Points.

---

## Overview of Kernel Components and Interfaces

The OS2KRNL modules lies at the heart of OS/2 - it is essentially operating system.

The kernel comprises an number of internal components, each responsible for a different aspect of running the system. It also has a number of interfaces that provides services to applications, device drivers and file systems.

These aspects are now considered in a little more detail and are summarised in the diagram shown under: [The OS/2 Kernel's Interfaces \(steady state\)](#).

---

## Kernel Components

Task management and the Scheduler.

This is responsible for thread and process management. The functions performed include:

Thread and Process creation and termination.

Thread scheduling (priority and state management).

Preparing threads for dispatching.

Blockin and Running.

Implementing the Thread and Process related APIs.

The Scheduler's principle control blocks are:

PTDA Per Task Data Area

TCB Thread Control Block

TSD Thread Swappable Data

TSS Task State Segment (H/W)

System Loader

This is responsible for load module management. The Loader's principle responsibilities include:

Bringing modules into memory and performing fixups.

- Managing modules resources.
- Managing dynamic linking.
- Tracking module references.
- Deleting modules from memory.
- Managing the discarding and swapping of module pages.
- Implementing module related APIs.

The Loader's principle control blocks are:

MTE	Module Table Entry
SMTE	Swappable Module Table Entry
OTE	Object Table Entry
STE	Segment Table Entry

## Memory Management

Memory Management is responsible for managing physical, virtual, and swapper memory. It's principle roles include:

- Allocation and assignment of physical pages of memory.
- Allocation and assignment of virtual storage.
- Managing the swapper.
- Memory locking.
- Implementing memory related APIs.

The principle control blocks of Memory Management are:

VMAR	Virtual Memory Arena Record
VMOB	Virtual Memory Object Record
PF	Page Frame Structure
VP	Virtual Page Structure
PTE	Page Table Entry (H/W)

## File System

The File System kernel component responsibilities include:

- Access to FAT formatted media.
- Interfacing with File System Drivers for accessing non-FAT media.
- Managing and tracking the status of all open files.
- Path Management
- File sharing and serialisation.
- Providing helper Kernel services for FSDs.
- Implementation of all File System APIs.

The principle control blocks of the File System include:

MFT	Master File Table Entry
SFT	System File Table Entry
CDS	Current Directory Structure

FSC

File System Control Block

#### Device and I/O Management

This component is responsible for interfacing with Physical Device Drivers. Its responsibilities include:

Routing requests to PDDs from applications

Managing interrupts

Providing helper kernel services for PDDs.

The principle control blocks for device management include:

IRQI

IRQ Information Array

DIRQ

Device IRQ Information

REQ

PDD request Packet.

DEV

PDD device header.

#### Virtual Dos Machine

This component is responsible for providing the entire Dos Machine emulation. This has not been covered in this book, except for the Virtual Device Driver interface.

---

## Kernel Interfaces

The Kernel provides the following external interfaces:

#### Application (R3/2) Interface.

Application access kernel services via GDT call gates. These are called either directly from the application program or via the DOSCALL1.DLL module, where additional Ring 3 processing is required before calling the kernel. Some system interfaces are able to be implemented entirely within Ring 2/3. In these cases, DOSCALL1.DLL does not make any kernel calls.

The kernel interfaces are represented by a fictitious module called DOSCALLS.DLL.

#### File System Driver (FSD)

The FSDs run in ring 0 as separately loaded modules. They are provided a set of interfaces to the kernel via the FSD\_Hlp (File System Helper) calls.

#### Physical Device Driver (PDD)

The PDDs run in ring 0 as separately loaded modules. They are provided a set a interfaces to the kernel via the Dev\_Hlp (Device Helper) calls.

#### Virtual Device Driver (VDD)

The VDDs run in ring 0 as separately loaded modules. They are provided a set a interfaces to the kernel via the VDD\_Hlp (Virtual Device Driver Helper) calls.

#### Compatibility BIOS

The compatibility BIOS resides within the OS2LDR module. It provides a hardware implementation independent layer through which the kernel access the BIOS. The interface to the CBIOS from the kernel is provided by the Dos\_Hlp (Dos Helper Services). These are not available for access by PDDs, VDDs or FSDs, however a limited set of Dos\_Hlp calls are provided via the TESTCFG.SYS and OEMHLP\$ device drivers.

#### Notes:

OEMHLP\$ is not a separately loaded module - it is resident within the OS2LDR module.

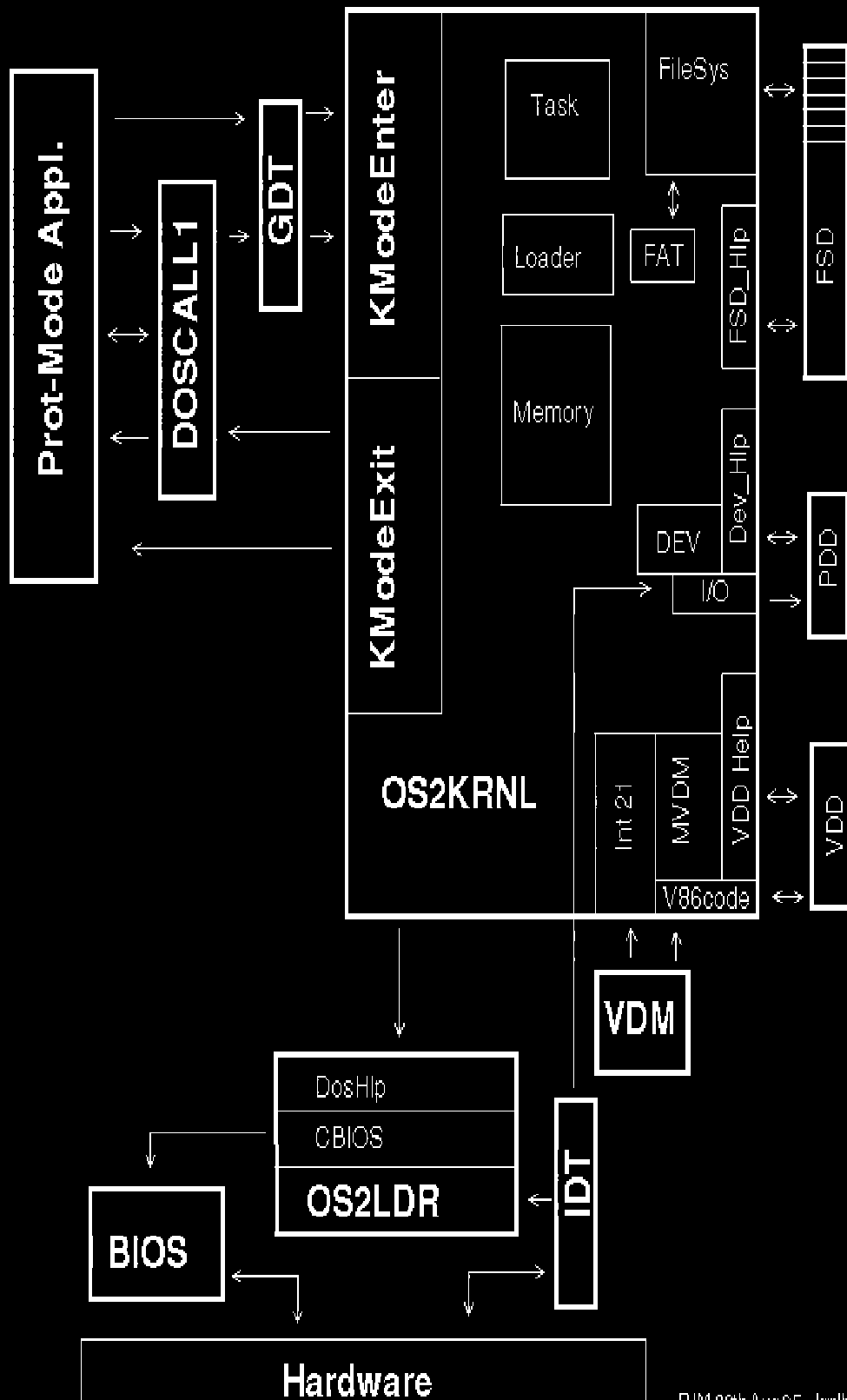
OS2LDR is responsible for loading the Kernel at system initialisation time. It does not get involved with the loading of Application Programs, PDDs, VDDs for FSDs during normal running - that function is performed by the System Loader component of the Kernel.

---



## The OS/2 Kernel's Interfaces

# The OS/2 Kernel's Interfaces (steady state)



---

## Miscellaneous System Control Block Reference

The following control blocks are described in this section:

[System Anchor Segment \(SAS\)](#)

[Block Management Package \(BMP\)](#)

[Record Management Package \(RMP\)](#)

An overview of the Miscellaneous System Control Blocks follows:

---

## Miscellaneous System Diagrams

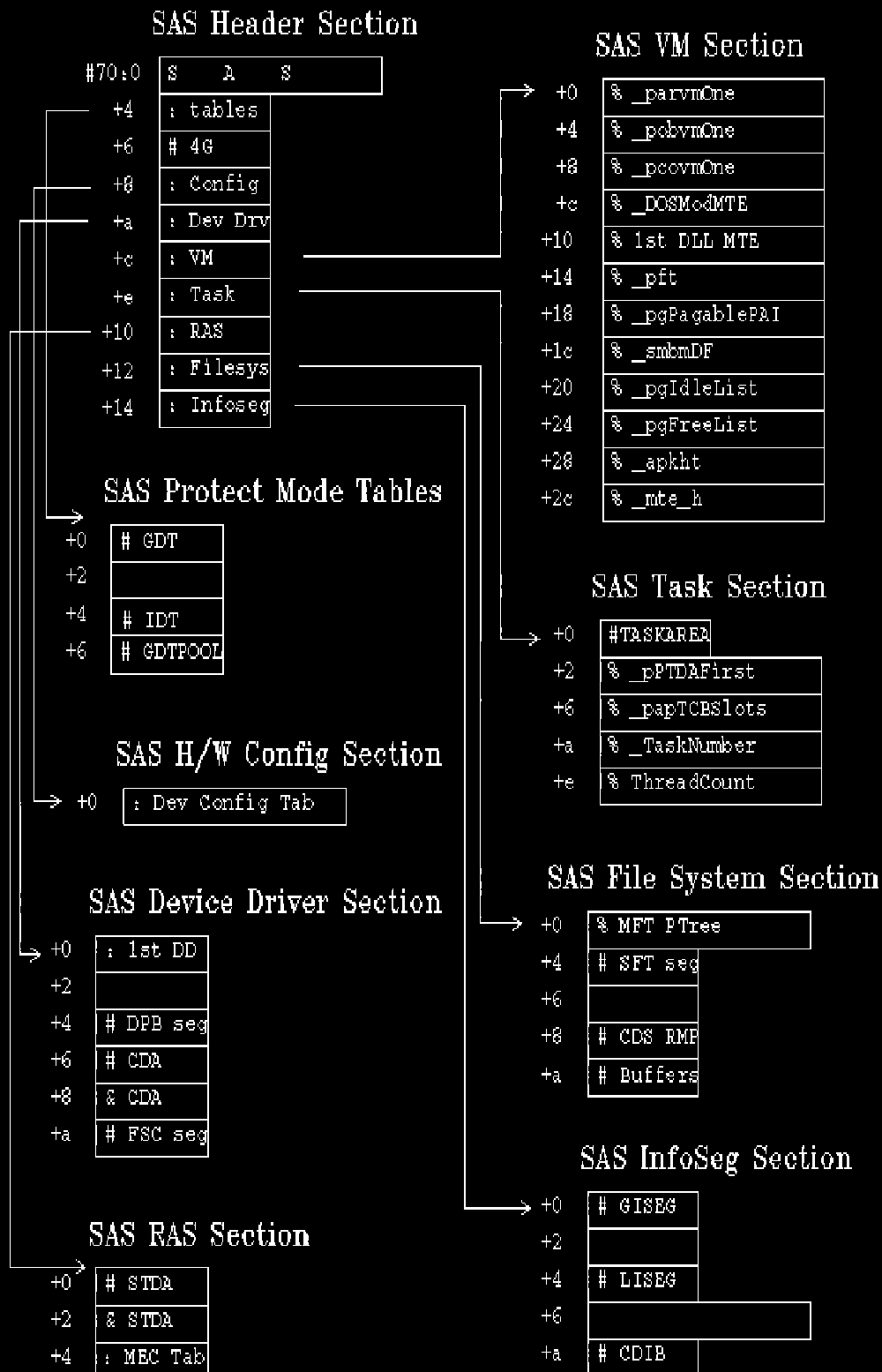
The following diagrams are illustrated:

[The System Anchor Segment](#)

---

## The System Anchor Segment

# The System Anchor Segment



# System Anchor Segment (SAS) for OS/2 Warp V4.0 and OS/2 Warp V3.0

The SAS is the common anchor for many system control blocks and control block chains.

## Pointers

**70:0** maps the SAS as a read-only segment.

**78:0** maps the SAS as a read/write segment.

## Locations

Built statically within the OS2KRNL load module.

## VM Owner

**os2krnl (0xffaa)**

## Format

**SAS Base Section.**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SAS_signature	+0	4	A	"SAS "
SAS_tables_data	+4	2	W	offset to tables section
SAS_flat_sel	+6	2	W	FLAT selector for kernel data
SAS_config_data	+8	2	W	offset to configuration section
SAS_dd_data	+a	2	W	offset to device driver section
SAS_vm_data	+c	2	W	offset to Virtual Memory section
SAS_task_data	+e	2	W	offset to Tasking section
SAS_RAS_data	+10	2	W	offset to RAS section
SAS_file_data	+12	2	W	offset to File System section
SAS_info_data	+14	2	W	offset to infoseg section

**SAS\_tables\_section** Protected Mode tables section.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SAS_tbl_GDT	+0	2	W	selector for GDT
SAS_tbl_LDT	+4	2	W	selector for LDT
SAS_tbl_IDT	+6	2	W	selector for IDT
SAS_tbl_GDTPOOL	+8	2	W	selector for GDTPOOL

**SAS\_config\_section** Configuration Section section.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SAS_config_table	+0	2	W	offset for Device Configuration Table

(DevConfigTbl)

#### **SAS\_dd\_section** Device Driver Section.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SAS_dd_bimodal_chain	+0	2	W	offset for the first bimodal device driver's device header
SAS_dd_real_chain	+2	2	W	offset for the address of the first real mode device driver's device header
SAS_dd_DPB_segment	+4	2	W	selector for Drive Parameter Block (DPB) segment
SAS_dd_CDA_anchor_p	+6	2	W	selector for ABIOS protected mode Common Data Area
SAS_dd_CDA_anchor_r	+8	2	W	segment for ABIOS real mode Common Data Area
SAS_dd_FSC	+a	2	W	selector for FSC

#### **SAS\_vm\_section** Virtual Memory Management section.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SAS_vm_arena	+0	4	D	Flat offset of arena records
SAS_vm_object	+4	4	D	Flat offset of object records
SAS_vm_context	+8	4	D	Flat offset of context records
SAS_vm_krnl_mte	+c	4	D	Flat offset of kernel MTE records
SAS_vm_glbl_mte	+10	4	D	Flat offset of global MTE linked list. Note this field points into the chain to pick up global MTEs only. Use SAS_vm_all_mte to find all the MTEs.
SAS_vm_pft	+14	4	D	Flat offset of page frame table
SAS_vm_prt	+18	4	D	Flat offset of page range table
SAS_vm_swap	+1c	4	D	Pointer to flat offset of swapper disk frame bit map followed by the size of the bit map in bits WARNING: the bit map offset and size are volatile
SAS_vm_idle_head	+20	4	D	Flat offset of Idle Head
SAS_vm_free_head	+24	4	D	Flat offset of Free Head
SAS_vm_heap_info	+28	4	D	Flat offset of Heap Array
SAS_vm_all_mte	+2c	4	D	Flat offset of all MTEs linked list

#### **SAS\_task\_section** Tasking section.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SAS_task_PTDA	+0	2	W	selector for current PTDA
SAS_task_ptdaptrs	+2	4	D	FLAT offset for process tree head
SAS_task_threadptrs	+6	4	D	FLAT address for TCB address array
SAS_task_tasknumber	+a	4	D	offset for current TCB number

SAS_task_threadcount	+e	4	D	offset for ThreadCount
----------------------	----	---	---	------------------------

#### SAS\_RAS\_section RAS section.

Field Name	Offset	Length	Type	Description
SAS_RAS_STDA_p	+0	2	W	selector for System Trace Data Area (STDA)
SAS_RAS_STDA_r	+2	4	D	segment for System Trace Data Area (STDA)
SAS_RAS_event_mask	+6	4	D	offset for trace event mask

#### SAS\_file\_section File System section.

Field Name	Offset	Length	Type	Description
SAS_file_MFT	+0	4	D	handle to MFT PTree
SAS_file_SFT	+4	2	W	selector for System File Table (SFT) segment
SAS_file_VPB	+6	2	W	selector for Volume Parameter Block (VPB) segment
SAS_file_CDS	+8	2	W	selector for Current Directory Structure (CDS) segment
SAS_file_buffers	+a	2	W	selector for buffer segment

#### SAS\_info\_section Information Segment section.

Field Name	Offset	Length	Type	Description
SAS_info_global	+0	2	W	selector for global info seg
SAS_info_local	+2	4	D	address of curtask local infoseg
SAS_info_localRM	+6	4	D	address of DOS task's infoseg
SAS_info_CDIB	+a	2	W	selector for Codepage Data

## Block Management Package Header (VMBH) for OS/2 Warp V4.0 and OS/2 Warp V3.0

The [BMP](#) is a generalised facility used to manage tables of fixed length entities. The BMP consists of a header followed by the table it manages. The use of the BMP is many and varied, but almost always occurs where an expandable table of fixed length entries is required by the system.

#### Pointers

The **VMBH** prefixes tables, which are pointed to by:

**\_parVMOne**

The table of VM Arena Records (**VMARs**).

<b>_pobVMOne</b>	The table of VM Object Records ( <b>VMOBs</b> ).
<b>_paVMAliases.</b>	The table of VM Alias Records ( <b>VMALs</b> ).
<b>_pcoVMOne</b>	The table of VM Context Records ( <b>VMCO</b> ).
<b>s2BmpSel</b>	BMP Selector for Device Driver Strategy 2 Request Packets.
<b>pFSLIST_BMP</b>	ListIO BMP segment.
<b>NmpBmpSel</b>	BMP Selector for Named Pipe NP Structures.
<b>_pVPB_BMP</b>	Volume Parameters Block BMP.
<b>_pFPEMVBH</b>	Floating point emulator BMP.
<b>_pkshdVMDescs</b>	BMP for Swappable Kernel Heap Descriptors ( <b>VMKSHD</b> ).
<b>pslhLockHandles</b>	BMP for Memory Lock Handles.
<b>_pbhPwvRPD</b>	PerfView counters
<b>_paPTDA</b>	The array of <b>PTDA</b> structures.
<b>_paTCB</b>	The array of <b>TCB</b> structures.

#### Locations

Many.

#### VM Owner

The VMBH is usually part of the the object from which the BMP is allocated, thus adopts the object id of the table.

#### Format

**VMBH** BMP Header Structure.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
bh_pbFreeHead	+0	4	D	Free list pointer
bh_pbEndBlocks	+4	4	D	End of valid blocks
bh_pbEndVirt	+8	4	D	End of virtual memory
bh_pbLastBusy	+C	4	D	Pointer to last busy block
bh_pfnisbusy	+10	4	D	Busy block identifier function
bh_filler	+14	4	D	Paragraph boundary filler
bh_flgpgtype	+18	4	D	New page type flags
bh_cbPerBlock	+1c	2	W	Size of a block in bytes
bh_hob	+1e	2	W	Object record handle

**bh\_flgpgtype** flag definitions.

Name	Bit Mask	Description
PG_CONTIG	0x00000001	contiguous physical memory
PG_NOINCR	0x00000001	don't increment physical addrs
PG_W	0x00000002	Writable - value from pte
PG_U	0x00000004	user mode accessible - from pte
PG_X	0x00000008	eXecutable



PG_R	0x00000010	Readable
PG_1M	0x00000020	must reside below 1 meg physical
PG_GUARD	0x00000040	guard page - from pte
PG_16M	0x00000040	must reside below 16 meg physical
PG_ZEROFILL	0x00000080	zero initialize pages
PG_SWAPONWRITE	0x00000100	value from vp
PG_UVIRT	0x00000200	value from pte
PG_RESIDENT	0x00000400	value from pte
PG_DISCARDABLE	0x00000800	value from vp

-----

## Record Management Package (RMP) for OS/2 Warp V4.0 and OS/2 Warp V3.0

The RMP is used to manage tables of variable length entities. It appears in a number of situations, particularly those that required ASCII strings, such as file names, to be managed.

### Pointers

**rp\_selector** of the RMP handle maps the RMP segment.

### Locations

RMP handles are located at the following labels:

<b>CharDevRMPRec</b>	Character Device Drivers
<b>SpoolDevRMPRec</b>	Spooler Device Drivers
<b>NmpRmpHand</b>	Named Pipes
<b>hDiscSegRmpStruc</b>	Discardable Segments
<b>ShareRmpStruc</b>	Named Shared Memory
<b>SysSemRmpHdl</b>	System Semaphores

### VM Owner

<b>CharDevRMPRec</b>	<b>chardevrmp (0xff35)</b>
<b>SpoolDevRMPRec</b>	<b>spldevrmp (0xff34)</b>
<b>NmpRmpHand</b>	<b>npipenpn (0xff30)</b>
<b>hDiscSegRmpStruc</b>	<b>discard (0xff6c)</b>
<b>ShareRmpStruc</b>	<b>mshrmp (0xff83)</b>
<b>SysSemRmpHdl</b>	<b>syssemrmp (0xff36)</b>

### Format

**rbhdr** RMP Header Structure.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	---------------	---------------	-------------	--------------------

rb_size	+0	2	W	total size of segment
rb_free_size	+2	2	W	amount of free space
rb_1st_free	+4	2	W	link to first free block in seg
rb_last_free	+6	2	W	start of last free block
rb_hkh	+8	4	D	heap handle
rb_flags	+c	4	D	PG alloc/realloc flags
rb_hobowner	+10	2	W	hobowner
rb_hobmte	+12	2	W	hobmte
rb_first	+14	n	S	start of first record
rb_sz_field	+n+0	2	W	size of 'record size field'
	+n+2	n-2	S	record data

#### rbfree RMP Free Record Structure.

Field Name	Offset	Length	Type	Description
rf_size	+0	2	W	free block size (high bit set)
rf_prev_free	+2	2	W	link to prev free block in seg
rf_next_free	+4	2	W	link to next free block in seg

#### rparm RMP Handle Structure.

Field Name	Offset	Length	Type	Description
rp_flags	+0	1	B	flags
	+1	1	B	unused
rp_selector	+2	2	W	GDT selector to use

#### rp\_flags flag definitions.

Name	Bit Mask	Description
RPF_BUSY	0x01	Segment busy flag
RPF_WAITING	0x02	Somebody waiting flag
RPF_ALLOC	0x04	Segment allocated flag

## Semaphore Control Block Reference

The following control blocks are described in this section:

[FastSafeRamSemStruc](#)

[FastSafeRamSemStruc PM version](#)

[MuxTableEntry](#)

[RamSemStruc](#)

[Kernel Semaphore Structures](#)

[32-bit Semaphore Structures](#)

[System Semaphore Structures](#)

[PM/GRE Semaphore](#)

# FastSafeRamSemStruc

Pointers	TCB_SemInfo points to fs_RAMSem
Locations	Multiple, in user storage.
VM Owner	Multiple user storage owners.

Format

Field Name	Off	Length	Type	Description
FastSafeRamSemStruc	-a	e	S	Fast Safe Ram Semaphore
fs_Length	-a	2	W	Length of this structure
fs_ProcID	-8	2	W	Process ID of owner or zero
fs_ThrdID	-6	2	W	Thread ID of owner or zero
fs_Usage	-4	2	W	reference count
fs_Client	-2	2	W	16 bit field for use by owner
fs_RAMSem	+0	4	S	OS/2 RAM Semaphore

# FastSafeRamSemStruc PM Version

Pointers	TCB_SemInfo points to fs_RAMSem
Locations	Multiple, in user storage.
VM Owner	Multipl user storage owners.

Format

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
FastSafeRamSemStruc	-e	12	S	PM Fast Safe Ram Semaphore
fs_Length	-e	2	W	Length of this structure
fs_ProcID	-c	2	W	Process ID of owner or zero
fs_ThrdID	-a	2	W	Thread ID of owner or zero
fs_Usage	-8	2	W	reference count
fs_Client	-6	2	W	16 bit field for use by owner
fs_Timeout	-4	4	D	Timeout value
fs_RAMSem	+0	4	S	OS/2 RAM Semaphore

-----

## MuxTableEntry

### Locations

At label **MuxTable** in system storage

### VM Owner

os2krnl (0xffaa)

### Format

<i>Field Name</i>	<i>Off</i>	<i>Len</i>	<i>Type</i>	<i>Description</i>
MuxTableEntry	+0	9	S	Mux Table Entry
MuxLink	+0	2	W	Selector Link to next entry. Used to chain entries for a MuxWait request
MuxThreadID	+2	2	W	Thread Slot ID of waiter
MuxType	+4	1	B	Semaphore type.
MuxSemID	+5	4	D	Mux Semaphore handle.

### MuxType flag definitions

<i>name</i>	<i>value</i>	<i>description</i>
MUXTYPE_CLEAR	0	the mux table entry is clear
MUXTYPE_SYSEM	1	the ID is a system sem address
MUXTYPE_RAMHANDLE	2	the ID is a ram sem handle:offset
MUXTRYE_RAMPHYS	3	the ID is a ram sem physical address
MUXTYPE_EVENTSEM	4	the ID for a 32-bit event sem

-----

## RamSemStruc

Pointers

TCB\_SemInfo

Locations

Multiple, in user storage.

VM Owner

Multipl user storage owners.

Format

Field Name	Off	Length	Type	Description
RamSemStruc	+0	4	S	Ram Semaphore
RamSemOwner	+0	1	B	Ownership flag
RamSemFlag	+1	1	B	Ram Semaphore flag bit field
RamSemID	+2	2	W	RamSem Block/Run ID low word

RamSemFlag definitions

name	value	description
RAMSEM_WAITING	0x01	a thread is waiting on the sem
RAMSEM_MUXWAITING	0x02	a thread is muxwaiting on the sem

Notes:

The high-order 4 bit of the **RamSemFlag** are used as an extended owner field (to cater for more than 512 threads).

Only kernel code sets the **RamSemOwner** field to a [thread slot number](#). Ring 3 **RamSems** have **0xff** value for an owned **RamSem**

KSEM Structures for OS/2 Warp V4.0 and .OS/2 Warp V3.0 ALLSTRICT kernel

For **KSEM** formats for other versions of OS/2 see:

[KSEM](#) for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

Locations

Multiple, either imbeded in system structres, for example PTDA, MFT, or dynamically allocated from the kernel heaps.

VM Owner

Imbedded KSEMs assume the Owner Id of the imbedding structure. Stand-alone KSEMs allocated from the kernel heaps use id: **ksem (0xff7e)**

Format

KSEMSHR Shared Kernel Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ks_Signature	+0	4	D	
ks_bFlags	+4	1	B	
ks_bType	+5	1	B	
ks_Owner	+6	2	W	
ks_cusPendingWriters	+8	2	W	
ks_cusNest	+a	2	W	
ks_cusReaders	+c	2	W	
ks_cusPendingReaders	+e	2	W	

#### **KSEMMTX MUTEX Kernel Semaphore**

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ksm_Signature	+0	4	D	
ksm_bFlags	+4	1	B	
ksm_bType	+5	1	B	
ksm_Owner	+6	2	W	
ksm_cusPendingWriters	+8	2	W	
ksm_cusNest	+a	2	W	

#### **KSEMEVT Event Kernel Semaphore**

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
kse_Signature	+0	4	D	
kse_bFlags	+4	1	B	
kse_bType	+5	1	B	
kse_Owner	+6	2	W	
kse_cusPendingWriters	+8	2	W	

Ksem flag definitions.

Name	Bit Mask	Description
KSEM_NOINTERRUPT	0x1	
KSEM_WRITER	0x2	
KSEM_DISPLAYID	0x4	
KSEM_NOBLOCK	0x8	

-----

## KSEM Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0

# RETAIL kernel

## KSEMSHR Shared Kernel Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ks_bFlags	+0	1	B	
ks_bType	+1	1	B	
ks_Owner	+2	2	W	
ks_cusPendingWriters	+4	2	W	
ks_cusNest	+6	2	W	
ks_cusReaders	+8	2	W	
ks_cusPendingReaders	+a	2	W	

## KSEMMTX MUTEX Kernel Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ksm_bFlags	+0	1	B	
ksm_bType	+1	1	B	
ksm_Owner	+2	2	W	
ksm_cusPendingWriters	+4	2	W	
ksm_cusNest	+6	2	W	

## KSEMEVT Event Kernel Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
kse_bFlags	+0	1	B	
kse_bType	+1	1	B	
kse_Owner	+2	2	W	
kse_cusPendingWriters	+4	2	W	

-----

## 32-bit Semaphore Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0 ALLSTRICT kernel

For **32-bit Semaphore** formats for other versions of OS/2 see:

[32-bit Semaphore](#) for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

### Pointers

**TCB\_SleepId** points to **SEVENT**, **PEVENT**, **SMUTEX**, **PMUTEX**, **SMUX** or **PMUX** when waiting on the semaphore.

PTDA field **pPrSemTbl** points to the private semaphore table, which is indexed by the semaphore handle.

**pShSemTbl** points to the shared semaphore table, which is indexed by the low-order word of the semaphore handle. Each entry is a pointer to a semaphore main structre.

PTDA field **pPrSemTbl** points to the per-process private semaphore table, which is indexed by the low-order word of the semaphore handle. Each entry is a pointer to a semaphore main structre.

**pShSemStrTbl** points to the table of **SEMTBLNODE** entries. Each of these points to a hashed chain of **SEMSTRNODE** structures.

**Note:** Names are hashed by treating each name as table of null padded ULONGs and successively adding.

## Locations

Structures are allocated from the kernel heaps.

## VM Owners

<b>SEVENT</b>	<b>semstruc (0xffc2)</b>
<b>PEVENT</b>	<b>semstruc (0xffc2)</b>
<b>SMUTEX</b>	<b>semstruc (0xffc2)</b>
<b>PMUTEX</b>	<b>semstruc (0xffc2)</b>
<b>SMUX</b>	<b>semstruc (0xffc2)</b>
<b>PMUX</b>	<b>semstruc (0xffc2)</b>
<b>OPENQ</b>	<b>semopenq (0xffbf)</b>
<b>MUXQ</b>	<b>semmuxq (0xffbe)</b>
<b>SEMRECORD</b>	<b>semrec (0xffc0)</b>
<b>SEMTBLNODE</b>	<b>semtable (0xffc3)</b>
<b>SEMSTRNODE</b>	<b>semtable (0xffc3)</b>
Semaphore name	<b>semstr (0xffc1)</b>

## Format

### SEVENT Shared Event Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenQ	+8	4	D	pointer to the open queue
pszName	+c	4	D	name of semaphore, null if anonymous
pulCreatAddr	+10	4	D	Address passed in by app during create
ulSig	+14	4	D	0x54564553 "SEVT"
ptcb	+18	4	D	ptcb of caller

### PEVENT Private Event Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	------------	---------------	-------------	--------------------



usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenCt	+8	2	W	number of opens
pulCreatAddr	+a	4	D	Address passed in by app during create
ulSig	+e	4	D	0x54564550 "PEVT"

#### **SMUTEX** Shared Mutex Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usRequestCt	+6	2	W	number of requests
usSlotNum	+8	2	W	slot number of the owning thread
usRequesterCt	+a	2	W	number of requesters
pOpenQ	+c	4	D	pointer to the open queue
pszName	+10	4	D	name of semaphore, null if anonymous
pulCreatAddr	+14	4	D	Address passed in by app during create
ulSig	+18	4	D	0x58544D53 "SMTX"

#### **PMUTEX** Private Mutex Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usRequestCt	+6	2	W	number of requests
usSlotNum	+8	2	W	slot number of the owning thread
usRequesterCt	+a	2	W	number of requesters
usOpenCt	+c	2	W	number of opens
pulCreatAddr	+e	4	D	Address passed in by app during create
ulSig	+12	4	D	0x58544D50 "PMTX"

#### **SMUX** Shared Mux Wait Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
cSemRec	+2	2	W	count of semaphore records
pSemRec	+4	4	D	array of semaphore record entries
usWaitCt	+8	2	W	number of threads waiting on the mux
pOpenQ	+a	4	D	pointer to the open queue

pszName	+e	2	W	name of semaphore, null if anonymous
pulCreatAddr	+10	4	D	Address passed in by app during create
ulSig	+14	4	D	0x58554D53 "SMUX"

#### **PMUX** Private Mux Wait Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
cSemRec	+2	2	W	count of semaphore records
pSemRec	+4	4	D	array of semaphore record entries
usWaitCt	+8	2	W	number of threads waiting on the mux
usOpenCt	+a	2	W	number of opens
pPTDA	+c	4	D	pointer to PTDA of creator
pulCreatAddr	+10	4	D	Address passed in by app during create
ulSig	+14	4	D	0x58554D50 "PMUX"

#### **OPENQ** Open Queue Node Structure

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pidOpener	+0	2	W	process id of opening process
usOpenCt	+2	2	W	number of Opens for this process
pNextOpen	+4	4	D	pointer to next node in list
ulSig	+8	4	D	0x514E504F "OPNQ"

#### **MUXQ** Mux Queue Node Structure

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pMux	+0	4	D	pointer to a mux (shared or private)
pNextMux	+4	4	D	pointer to next mux waiter in list
ulSig	+8	4	D	0x5158554D "MUXQ"

#### **SEMRECORD** Semaphore Record Structure for MUX Wait Semaphores.

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
hsemCur	+0	4	D	semaphore handle
ulUser	+4	4	D	user value

#### **SEMSTRNODE** Semaphore String Node

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
hsem	+0	4	D	semaphore handle

psz	+4	4	D	pointer to the string
pNext	+8	4	D	pointer to next string node
ulSig	+c	4	D	0x444F4E53 "SNOD"

#### **SEMTBLNODE** Semaphore String Node Table Entry

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ulKey	+0	4	D	hash key
pStrNode	+4	4	D	pointer to string node

#### **usFlags** field definitions:

Name	Bit Mask	Description
DE_POSTED	0x0040	The event sem APIs set this flag if the event is in the posted state
DM_OWNER_DIED	0x0080	The process died while owning the mutex semaphore
DMW_MTX_MUX	0x0100	The muxwait semaphore APIs set this flag if the mux contains mutex sems
DHO_SEM_OPEN	0x0200	dh_OpenEventSem sets this flag to indicate that device drivers have opened the given semaphore
DE_16BIT_MW	0x0400	Part of a 16-bit MuxWait if this flag is set

-----

## 32-bit Semaphore Structures for OS/2 Warp V3.0 RETAIL kernel

#### **SEVENT** Shared Event Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenQ	+8	4	D	pointer to the open queue
pszName	+c	4	D	name of semaphore, null if anonymous
ptcb	+10	4	D	ptcb of caller

#### **PEVENT** Private Event Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usPostCt	+6	2	W	number of posts
pOpenCt	+8	2	W	number of opens

#### **SMUTEX** Shared Mutex Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usRequestCt	+6	2	W	number of requests
usSlotNum	+8	2	W	slot number of the owning thread
usRequesterCt	+a	2	W	number of requesters
pOpenQ	+c	4	D	pointer to the open queue
pszName	+10	4	D	name of semaphore, null if anonymous

#### **PMUTEX** Private Mutex Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
pMuxQ	+2	4	D	pointer to the mux queue
usRequestCt	+6	2	W	number of requests
usSlotNum	+8	2	W	slot number of the owning thread
usRequesterCt	+a	2	W	number of requesters
usOpenCt	+c	2	W	number of opens

#### **SMUX** Shared Mux Wait Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes
cSemRec	+2	2	W	count of semaphore records
pSemRec	+4	4	D	array of semaphore record entries
usWaitCt	+8	2	W	number of threads waiting on the mux
pOpenQ	+a	4	D	pointer to the open queue
pszName	+e	2	W	name of semaphore, null if anonymous

#### **PMUX** Private Mux Wait Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
usFlags	+0	2	W	attributes

cSemRec	+2	2	W	count of semaphore records
pSemRec	+4	4	D	array of semaphore record entries
usWaitCt	+8	2	W	number of threads waiting on the mux
usOpenCt	+a	2	W	number of opens
pPTDA	+c	4	D	pointer to PTDA of creator

#### **OPENQ** Open Queue Node Structure

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pidOpener	+0	2	W	process id of opening process
usOpenCt	+2	2	W	number of Opens for this process
pNextOpen	+4	4	D	pointer to next node in list

#### **MUXQ** Mux Queue Node Structure

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pMux	+0	4	D	pointer to a mux (shared or private)
pNextMux	+4	4	D	pointer to next mux waiter in list

#### **SEMSTRNODE** Semaphore String Node

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
hsem	+0	4	D	semaphore handle
psz	+4	4	D	pointer to the string
pNext	+8	4	D	pointer to next string node

-----

## System Semaphore Structures

### Pointers

**SysSemRmpHdl** contains the selector that points the system semaphore names RMP.

### Locations

**SysSemDataTable** is the location of the global system semaphores table. Each entry is a **SysSemTblStruc** structure.

PTDA field **SysSemPTDATbl** is the location of the per-process semaphore table.

PTDA per-process semaphore contains byte-length enties, which are per-semaphore use counts.

The semaphore handle indexes both the per-process and global semaphore tables.

**SysSemHighTable** locates the table of **SysSemHighTableS** structures.

## VM Owner

**syssemrmp (0xff36)** for the RMP that contains the semaphore names.

Other global tables are owned by **os2krnl (0xffaa)**.

## Format

### SysSemHandleStruc System Semaphore Handle Structure

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SysSemHighWord	+0	2	W	0x8000 for sys sems
SysSemPTDAIndex	+2	2	W	Index into the PTDA open sem table

### SysSemTblStruc System Semaphore Table Structure

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SysSemOwner	+0	2	W	thread owning this semaphore
SysSemFlag	+2	1	B	system semaphore flag bit field
SysSemRefCnt	+3	1	B	number of references to this sys sem
SysSemProcCnt	+4	1	B	number of requests for this owner
SysSemPad	+5	1	B	pad byte to round structure up to word

### SysSemHighTableS System Semaphore Table Extension Structure.

This is an extension of the SysSemTblStruc that is put into high memory so we don't impact the low data segment. It is only used in protected mode during process/thread termination.

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SysSemPidOwner	+0	2	W	pid owner, the thread owner has died

### SysSemNameStruc System Semaphore Name table structure, managed by an RMP.

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SysSemPtr	+0	2	W	

### SysSemFlag flag field definitions:

Name	Bit Mask	Description
SYSSEM_WAITING	0x01	a thread is waiting on the sem
SYSSEM_MUXWAITING	0x02	a thread is muxwaiting on the sem
SYSSEM_OWNER_DIED	0x04	the process/thread owning the sem died
SYSSEM_EXCLUSIVE	0x08	indicates a exclusive system semaphore
SYSSEM_NAME_CLEANUP	0x10	name table entry needs to be removed
SYSSEM_THREAD_OWNER_DIED	0x20	the thread owning the sem died
SYSSEM_EXITLIST_OWNER	0x40	the exitlist thread owns the sem

---

## PM/GRE Semaphore Structure

### Locations

**pmsemaphores** locates the table of PM/GRE semaphores.

### VM Owner

PMMERGE.DLL **hmte**

### Format

#### GRESEM PM/GRE Semaphore

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
acIdent	+0	7	A	GRESEM or PMSEM
fcSet	+7	1	B	386 Actual Semaphore
ulProcessThread	+8	4	D	owner process and thread id (PTid)
ulNestedUseCount	+c	4	D	# of times same PTid has accessed sem
ulWaitingCount	+10	4	D	# of PTids waiting on semaphore
ulUseCount	+14	4	D	# of times semaphore has been used
ulEventHandle	+18	4	D	Event Handle Semaphore
ulCallerAddr	+1c	4	D	Semaphore Caller

---

## Memory Management Control Block Reference

The following control blocks are described in this section:

[Page Frame Structure \(PF\)](#)

[Physical Arena Information Structures \(PAI\)](#)

[Per Arena Page Table Data \(PGDATA\)](#)

[Memory Alias Record \(VMAL\)](#)

[Memory Arena Header \(VMAH\)](#)

[Memory Arena Record \(VMAR\)](#)

[Memory Arena Type \(VMAT\)](#)

[Memory Context Record \(VMCO\)](#)

[Memory Object Record \(VMOB\)](#)

[Virtual Page Structure \(VP\)](#)

[Kernel Heap Header \(VMKH\)](#)

[Kernel Resident Heap Structures \(VMKRH, VMKRHY, VMKRHS, VMKRHF, VMKRHB, VMKRHBA\)](#)

[Kernel Swappable Heap Structures \(VMKSH, VMKSHD, VMKSHB\)](#)

An overview of the Memory Management Control Blocks follows:

---

## Memory Management Control Block Diagrams

The following diagrams illustrate the relationships between various Memory Management control blocks:

[Virtual Address Space Regions \(OS/2 Warp V4.0\)](#)

[Virtual Address Space Regions \(OS/2 Warp V3.0\)](#)

[Virtual Address Space Regions \(OS/2 V2.11\)](#)

[Virtual Address Space Management](#)

[Private Arena Private Data](#)

[Private Arena Shared Data](#)

[Shared Global Data](#)

[Shared Arena Instance Data](#)

[Virtual/Physical Page Management - Backed Storage](#)

[Virtual/Physical Page Management - Swapped Storage](#)

[CS Alias of Shared Instance Data](#)

[Memory Alias in Multiple Processes](#)

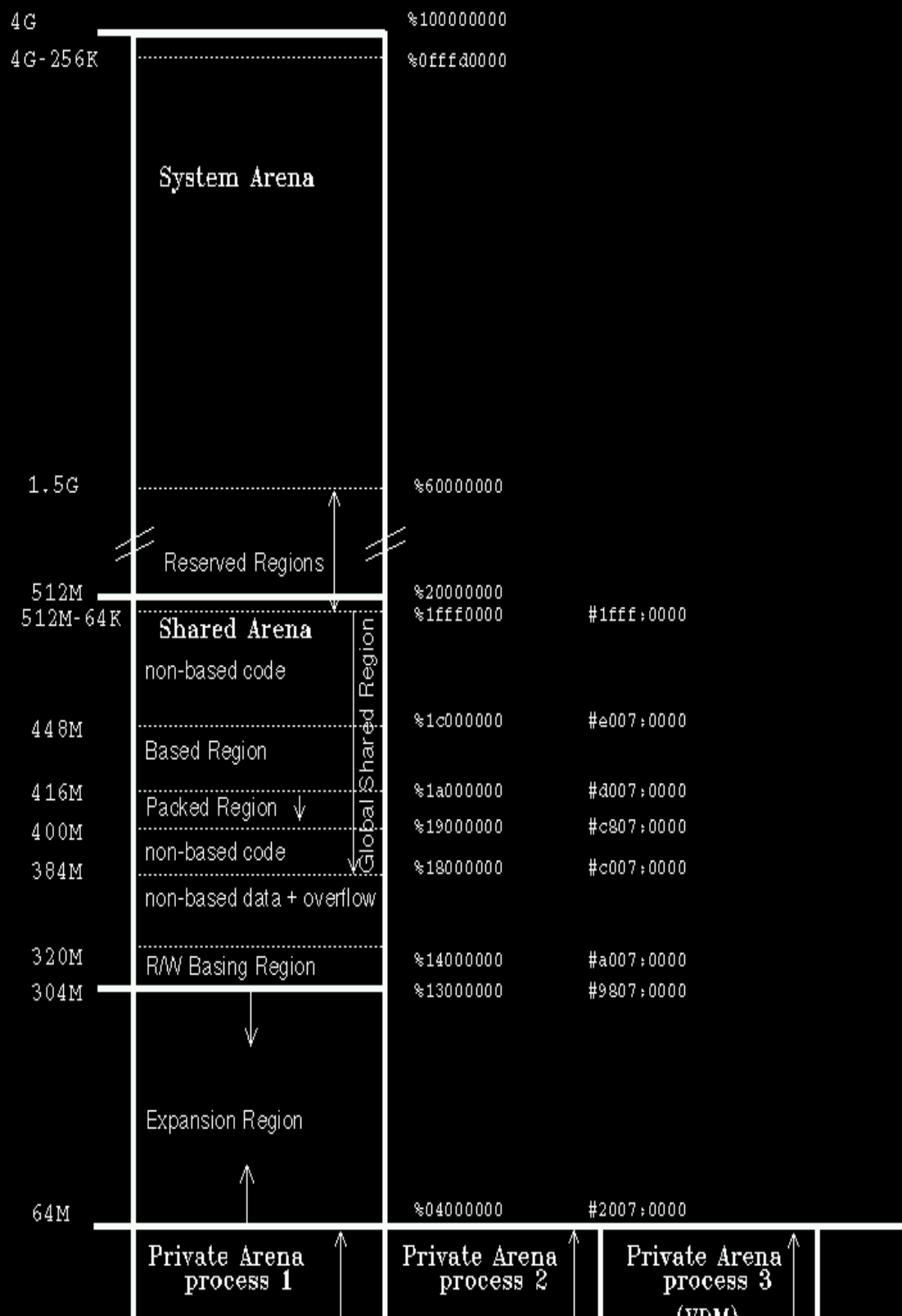
---

## Virtual Address Space Regions for OS/2 Warp V4.0 and OS/2 Warp V3.0 from fax pack 19



# Virtual Address Space Regions

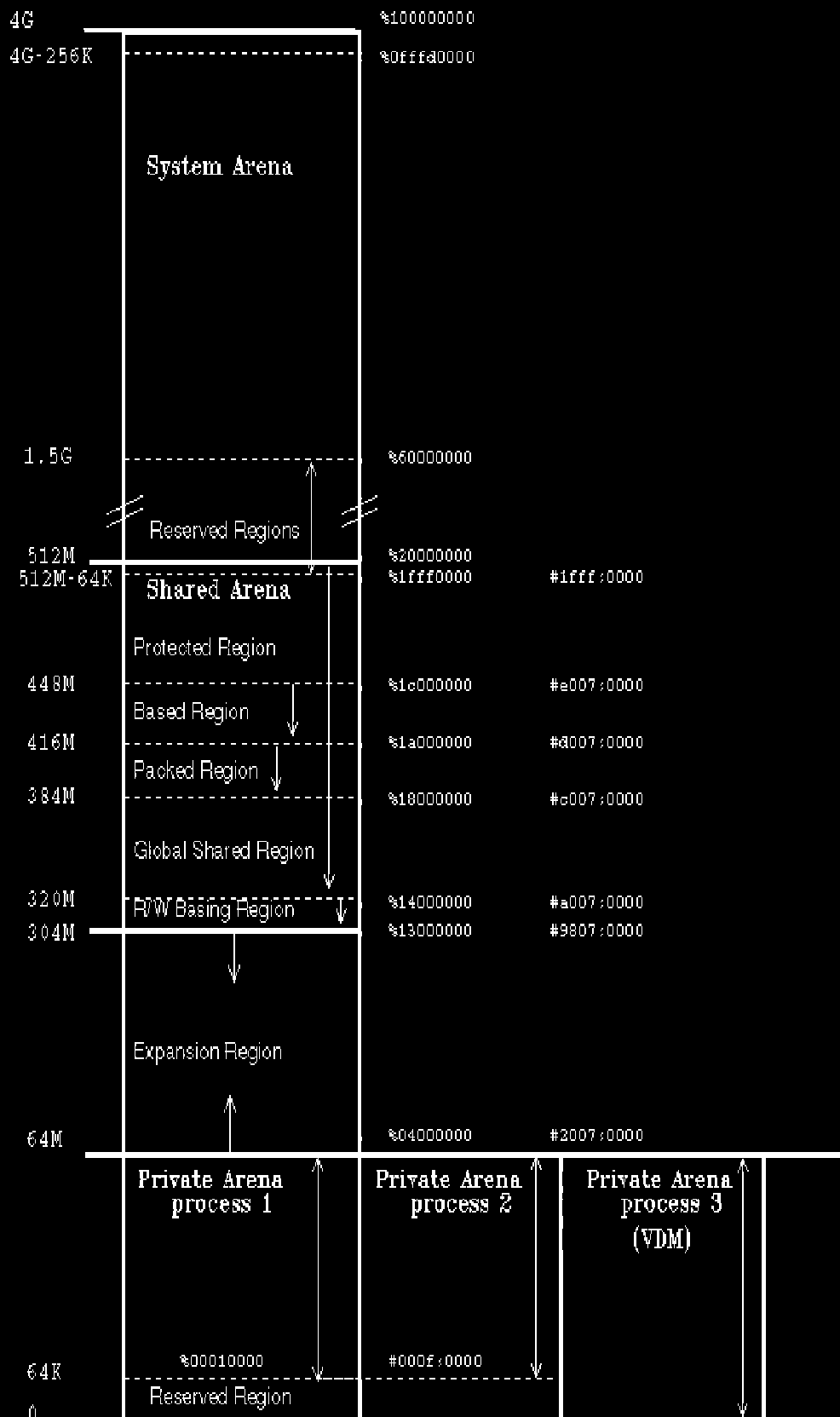
(OS/2 4.0 and 3.0 Fix Pack 19)



---

## Virtual Address Space Regions for OS/2 Warp V3.0

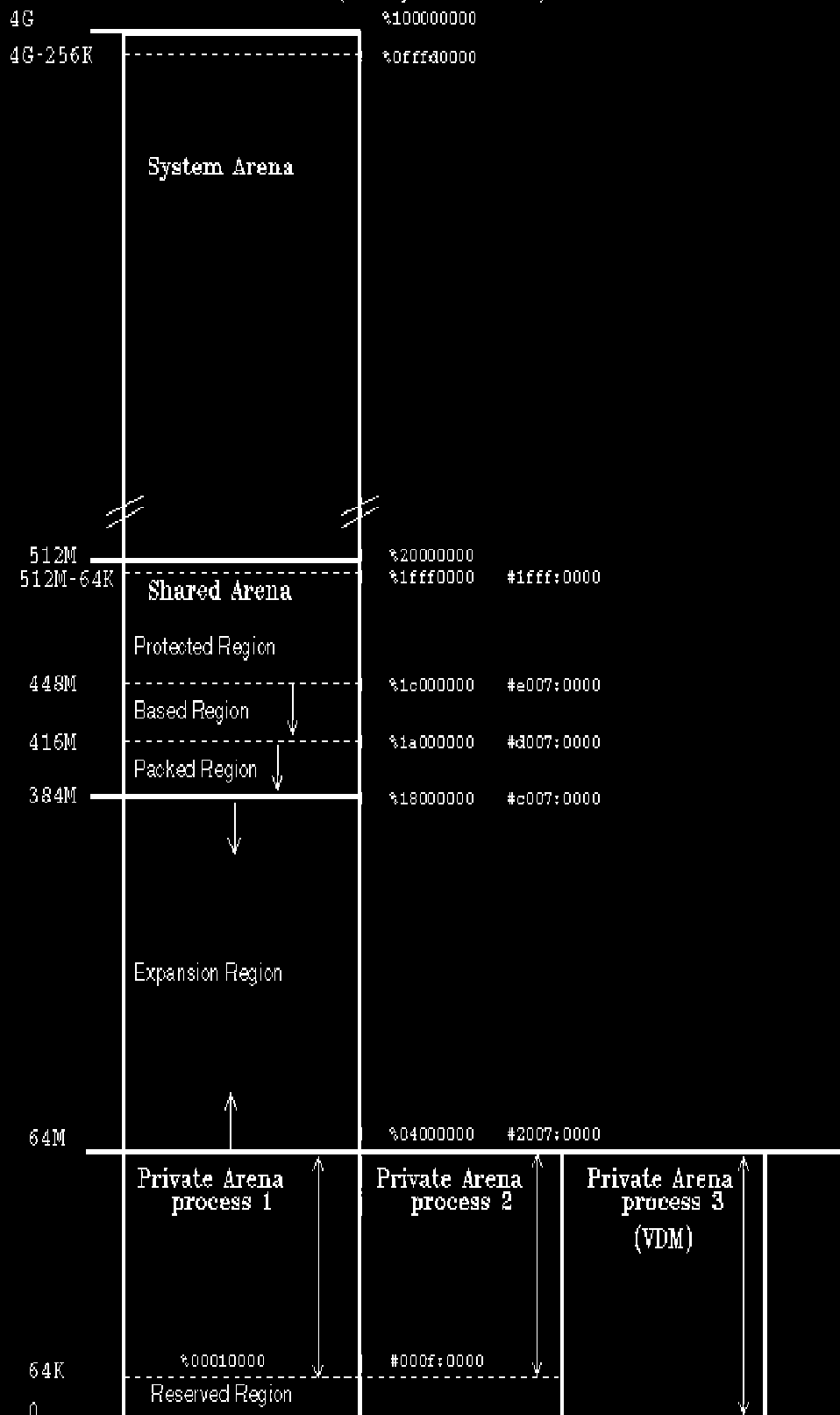
# Virtual Address Space Regions



---

## Virtual Address Space Regions for OS/2 V2.11

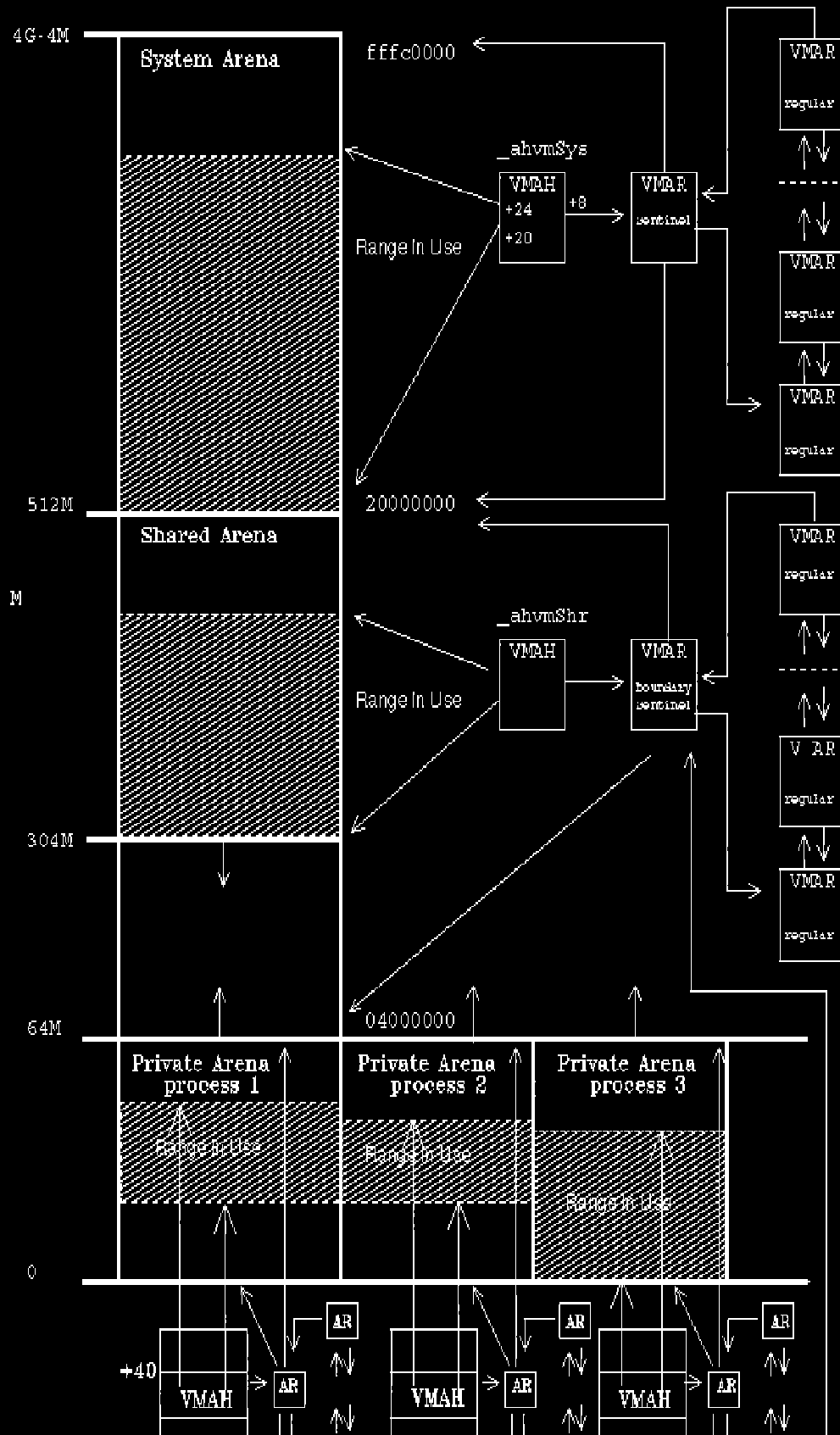
# Virtual Address Space Regions (OS/2 2.x)



---

## Virtual Address Space Management

# Virtual Address Space Management

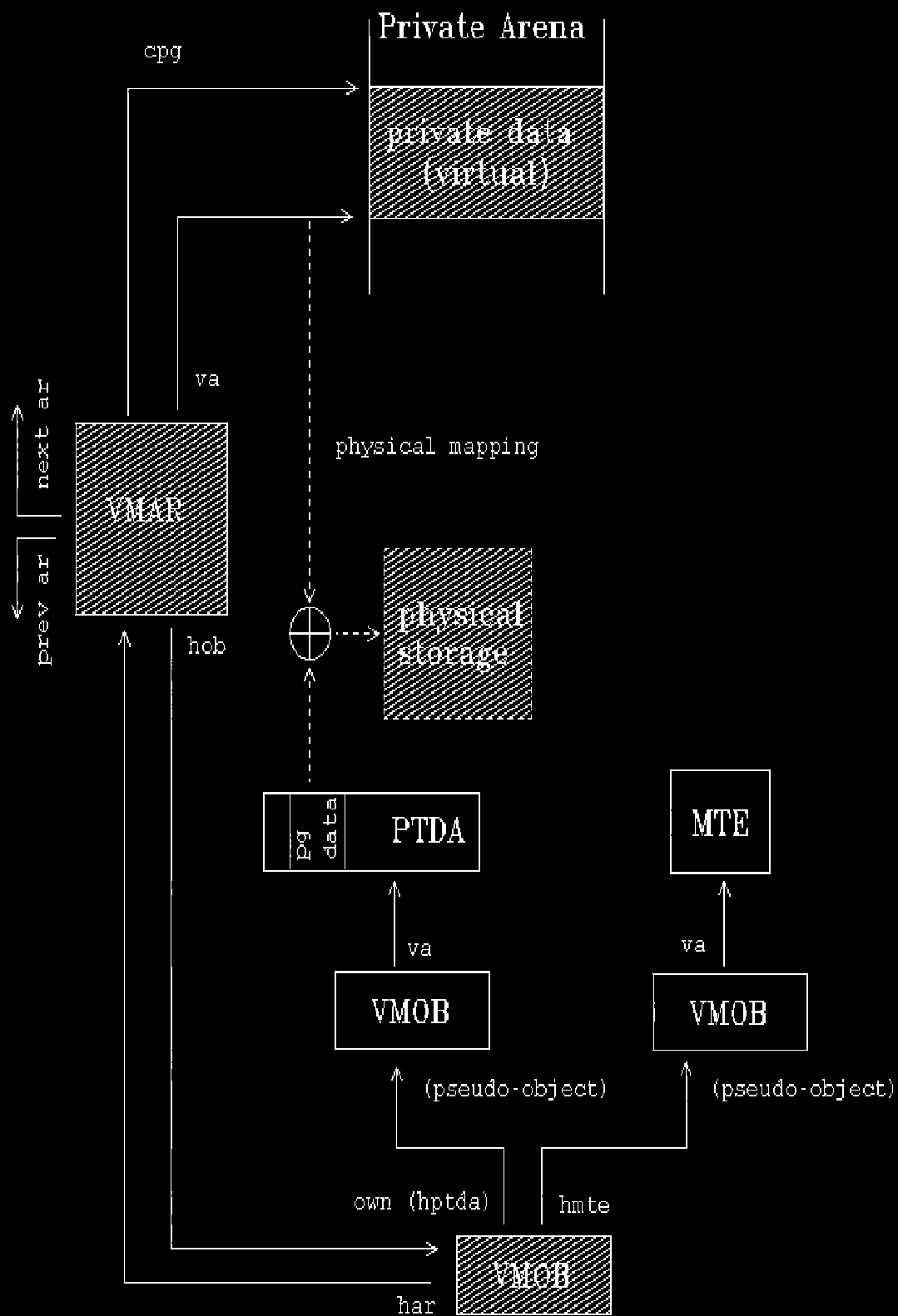


-----

Private Arena Private Data



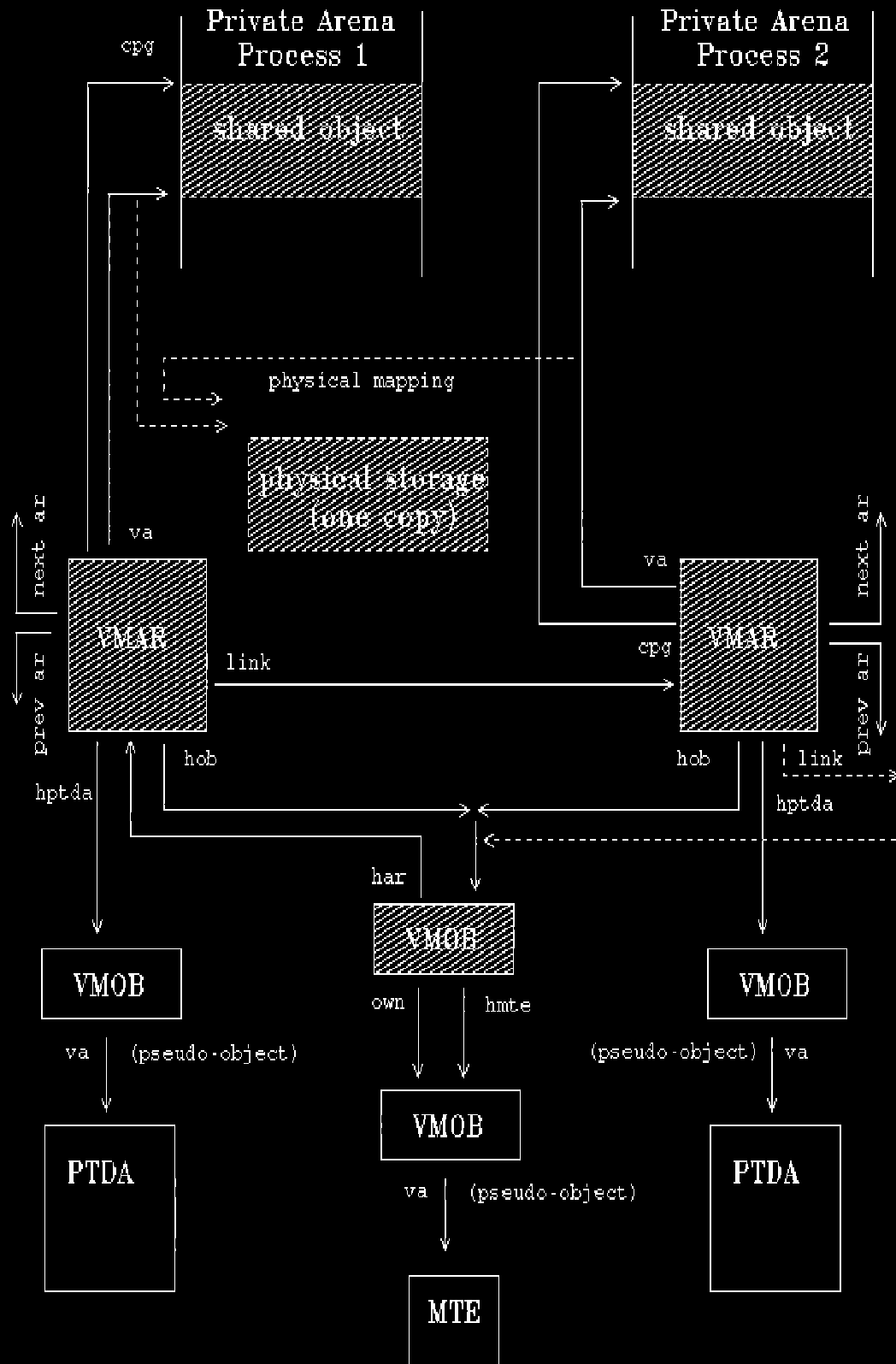
# Private Arena Private Data



---

# Private Arena Shared Data

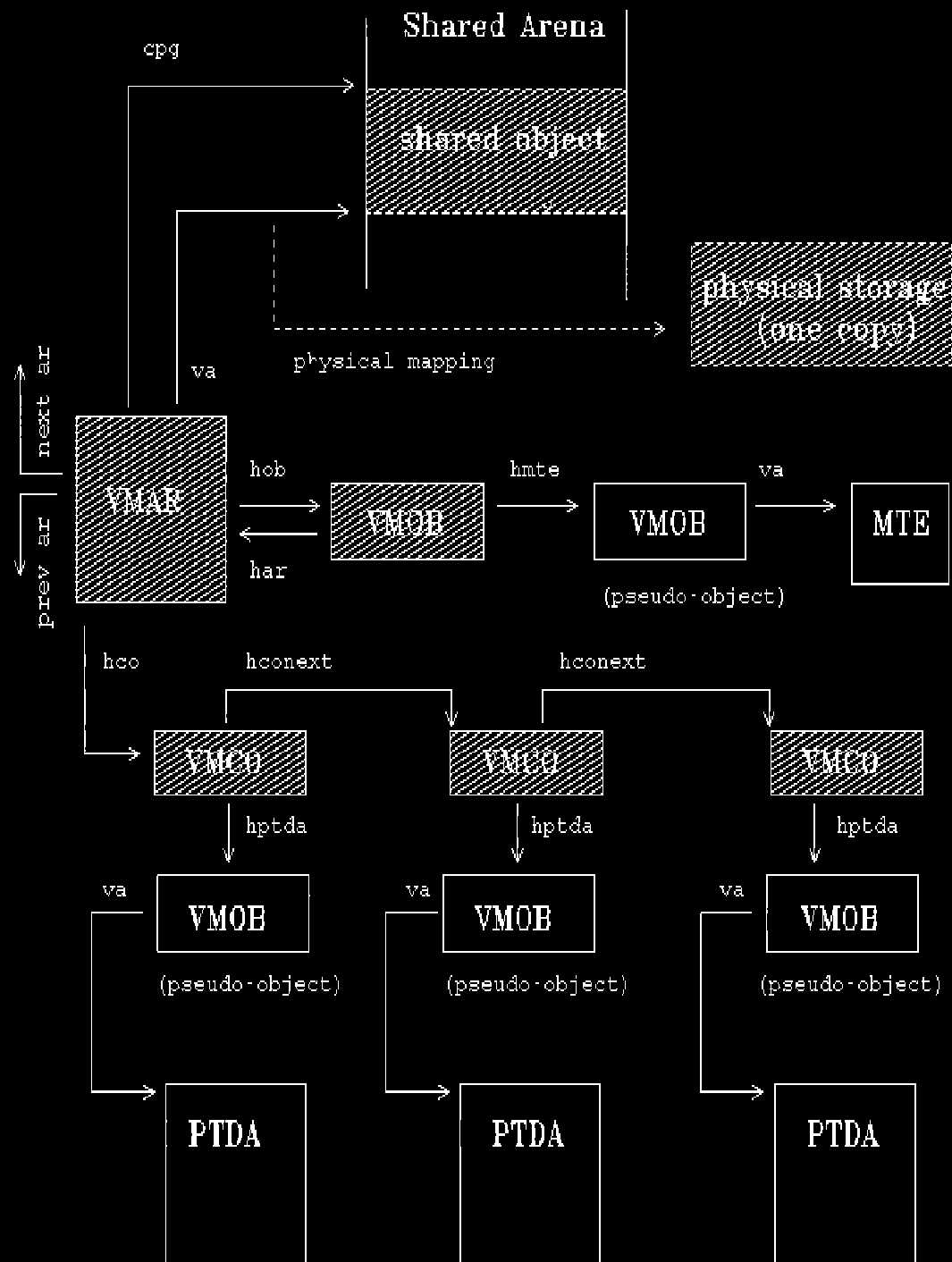
## Private Arena Shared Data



---

# Shared Global Data

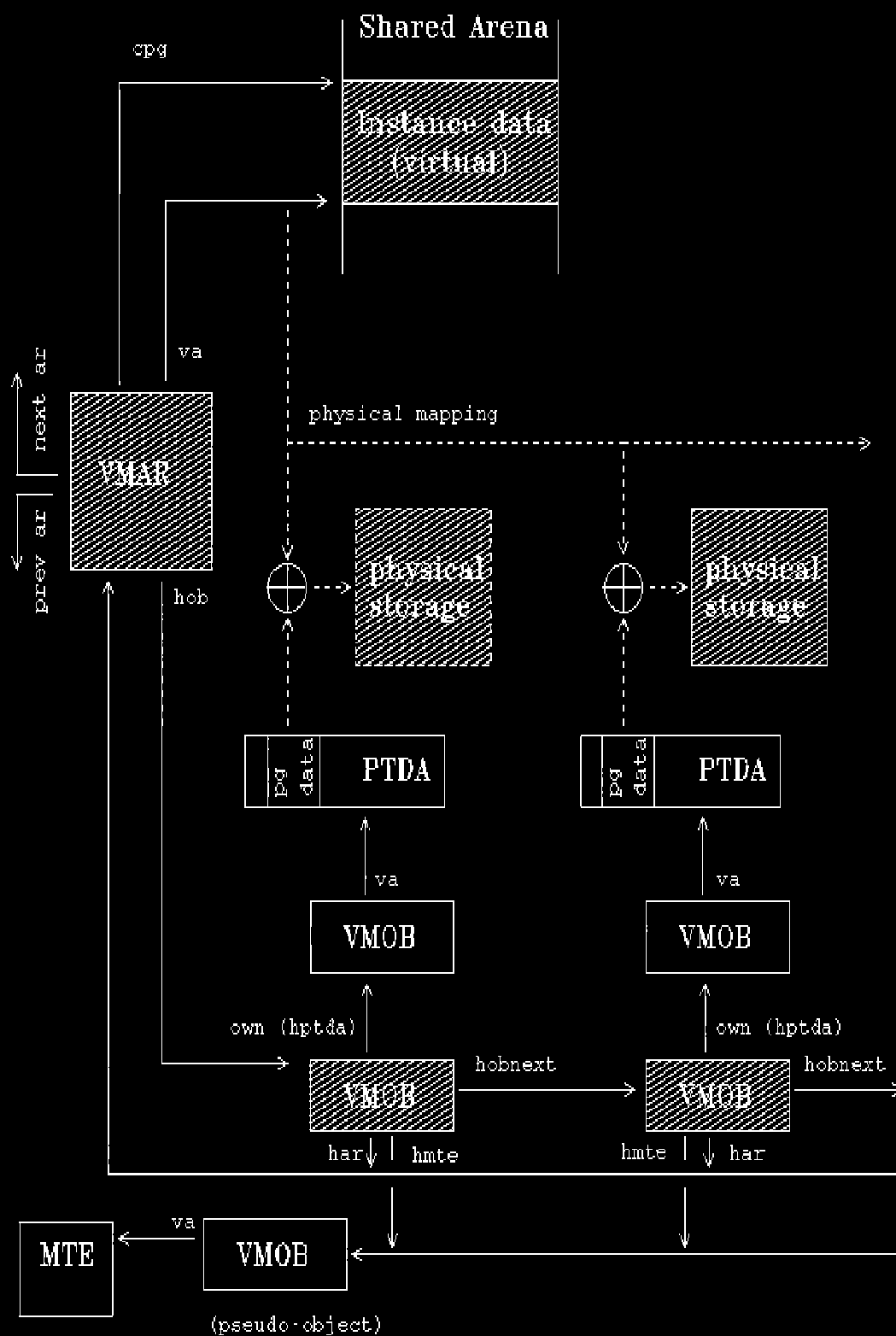
# Shared Global Data



---

# Shared Arena Instance Data

# Shared Arena Instance Data



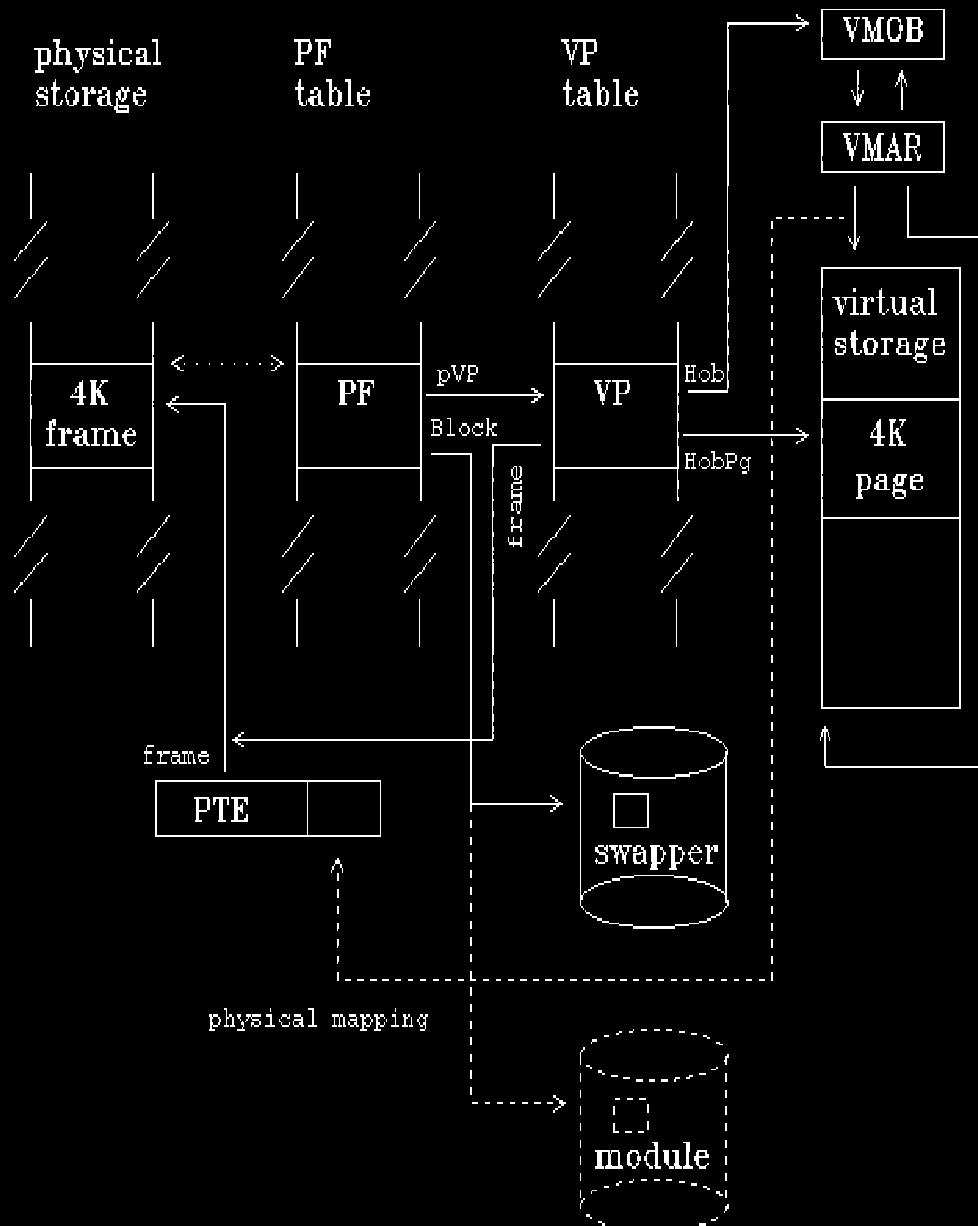
---

## Virtual/Physical Page Management - Backed Storage



# Page Management

## Backed Virtual Storage

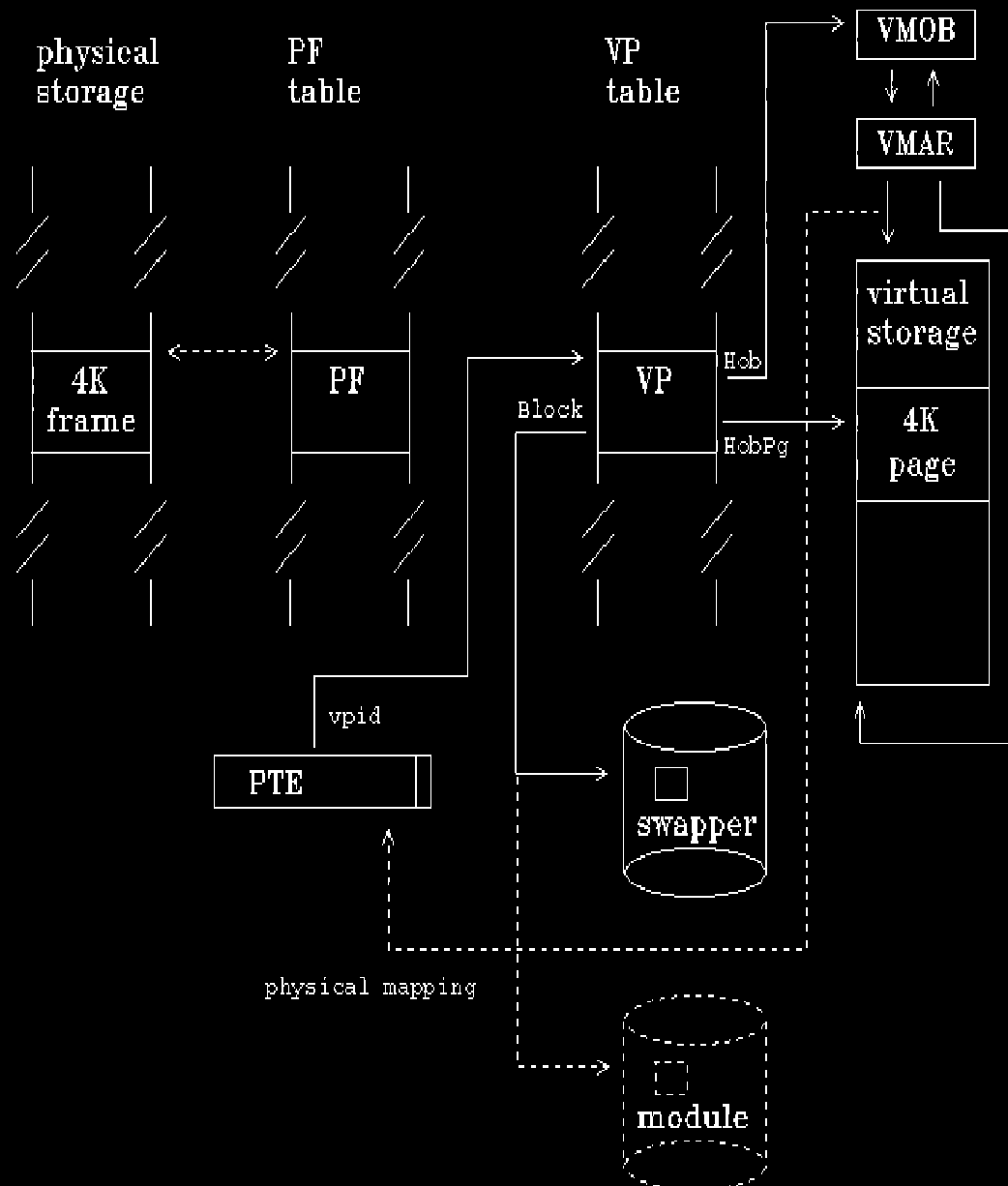


---

## Virtual/Physical Page Management - Swapped Storage

# Page Management

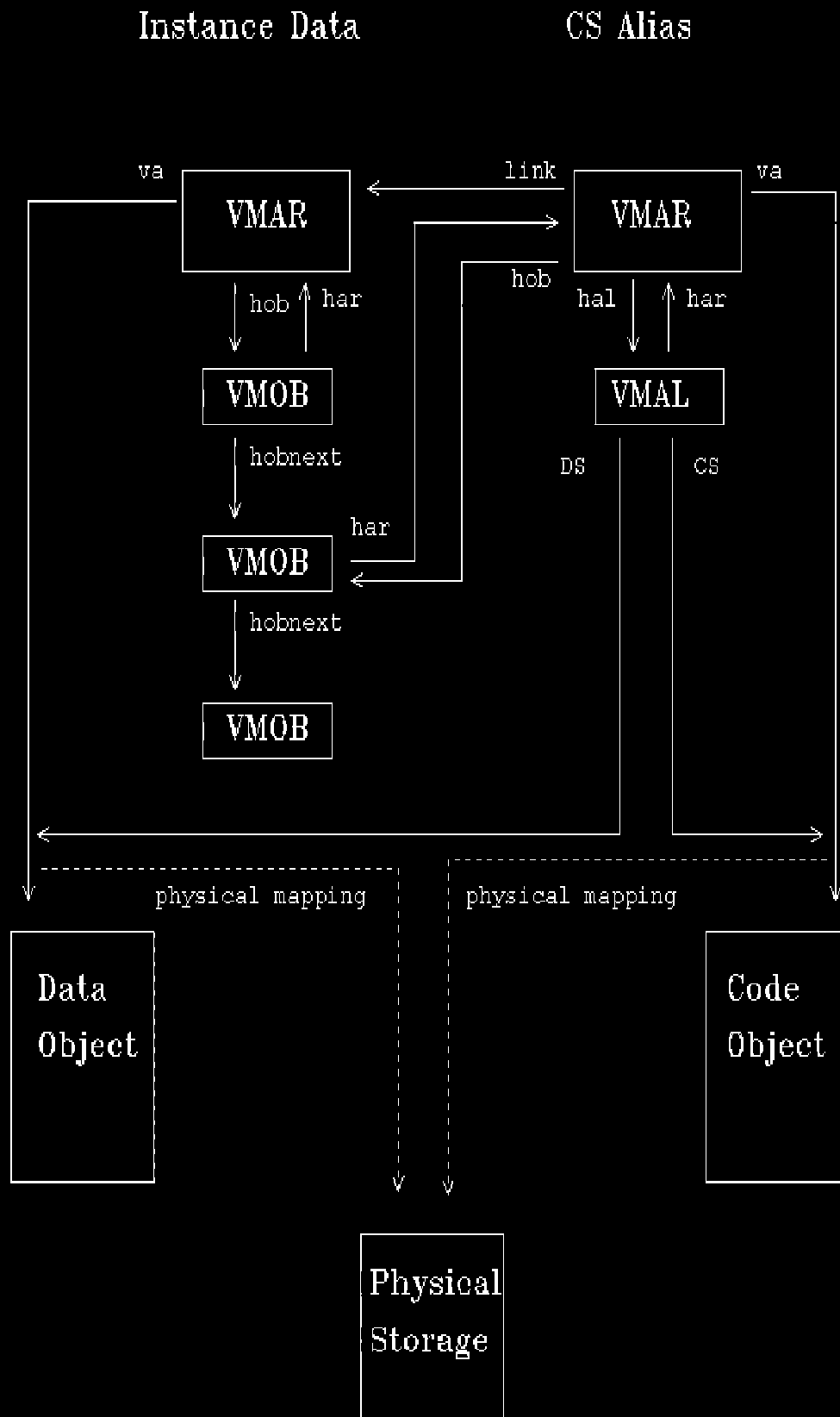
## Unbacked Virtual Storage



---

## CS Alias of Shared Instance Data

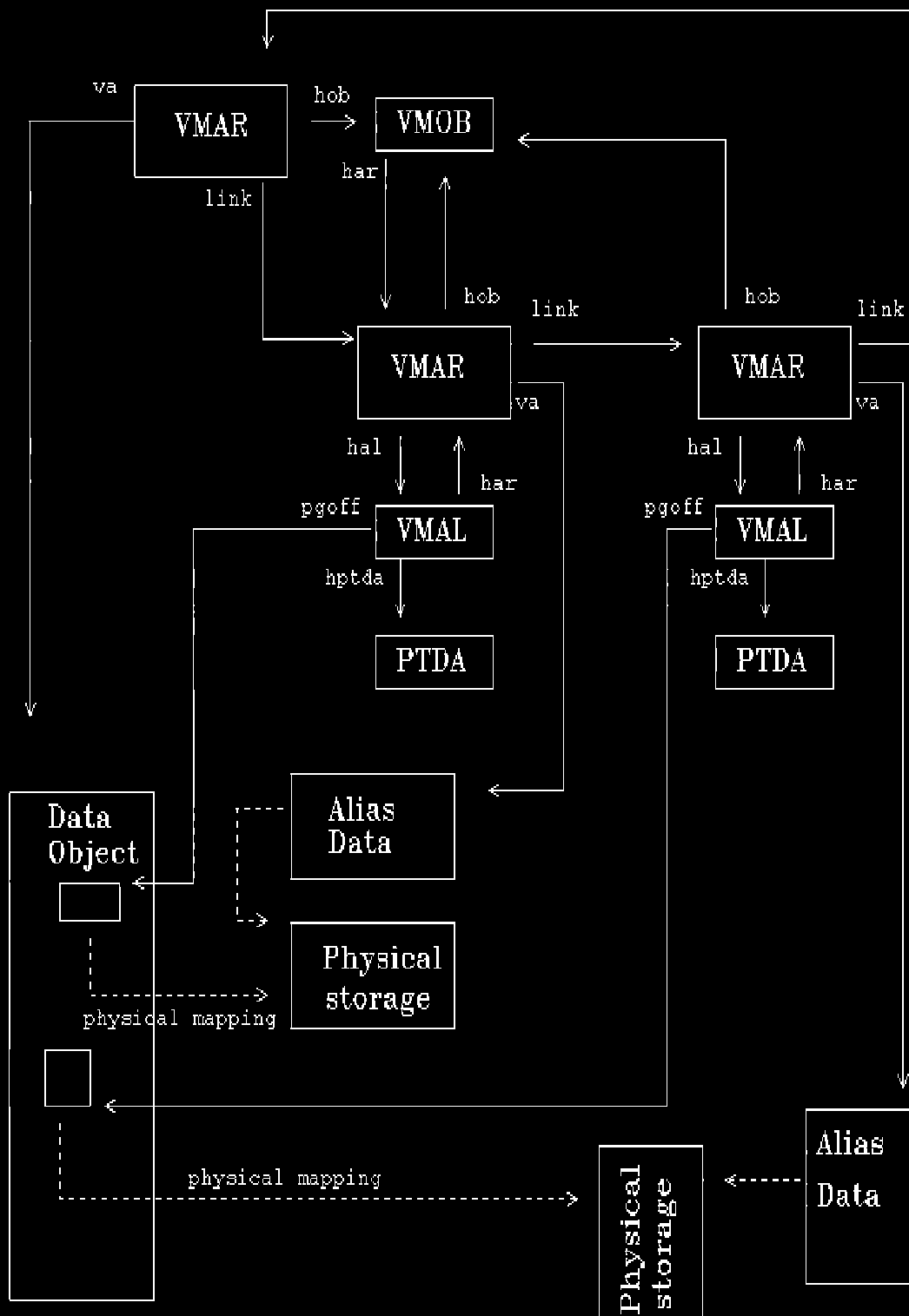
# CS Alias of Shared Instance Data



---

## Memory Alias in Multiple Processes

# Memory Aliases in Multiple Processes



-----

# Page Frame Structure

**Pointers**  
    \_pft points to the table of Page Frame Structres.

**Locations**  
    System Arena

**VM Owner**  
    pgpf (0xffb4)

**Format**

Field Name	Off	Length	Type	Description
apf_s	+0	c	S	active pf
pf_pvp	+0	4	D	vp cross link
pf_llock	+4	1	D	count of long term locks
pf_flags		0.4		flags
		0.4		pad
pf_refcount		2		count of ptes marked present
pf_block	+8	2.4	D	swp disk frame or ldr block number
		0.4		pad
pf_slock		1		count of short term locks
ipf_s	+0	c	S	idle page frame
pf_pvp	+0	4	D	vp cross link
pf_flink1	+4	1	D	forward link part 1 (low byte)
pf_flags		0.4		flags
vp_blink		2.4		backward link
pf_block	+8	2.4	D	swp disk frame or ldr block number
pf_flink2		1.4		forward link part 2 (high 1.4 bytes)
fpf_s	+0	c	S	free page frame
	+0	4	D	pad
pf_flink1	+4	1	D	forward link part 1 (low byte)
pf_flags		0.4		flags
vp_blink		2.4		backward link
	+8	2.4	D	pad
pf_flink2		1.4		forward link part 2 (high 1.4 bytes)

**pf\_flag** flag definitions:



<i>name</i>	<i>bit mask</i>	<i>description</i>
PF_FAST	0x1	frame is fast memory
PF_BUSY	0x2	frame is busy
PF_FREE	0x4	frame is free
PF_RES	0x8	reserved

-----

## Physical Arena Information Structures

### Pointers

**SAS\_vm\_prt**

### Locations

System Arena.

Two PAIs exist as part of the kernel load module. These are located at labels **\_pgPageablePAI** and **\_pgResidentPAI**.

### VM Owners

**os2krnl (0xffaa)**

### Format

**PAI** Physical Arena Information Structure.

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pai_pprt	+0	4	D	pointer to page range table
pai_nranges	+4	4	D	number of ranges in range table
pai_1M	+8	c	S	1M boundary structure
pb_frame	+8	4	D	last frame before boundary
pb_ppf	+c	4	D	last pf before boundary
pb_ppr	+10	4	D	page range containing boundary
pai_16M	+14	c	S	16M boundary structure
pb_frame	+14	4	D	last frame before boundary
pb_ppf	+18	4	D	last pf before boundary
pb_ppr	+1c	4	D	page range containing boundary
pai_end	+20	c	S	end of memory boundary structure
pb_frame	+20	4	D	last frame before boundary
pb_ppf	+24	4	D	last pf before boundary
pb_ppr	+28	4	D	page range containing boundary

**pagerange\_s** Page Range Table Entry.

Field Name	Off	Length	Type	Description
pri_lastframe	+0	4	D	last valid page in range
pri_firstframe	+4	4	D	first valid page in range

# Per Arena Page Table Data for OS/2 Warp V4.0 and OS/2 Warp V3.0

For **PGDATA** formats for other versions of OS/2 see:

[PGDATA](#) for OS/2 Warp 3.0 SMP

**Pointers**

For private arenas, **ptda\_ppgdata**.

**Locations**

System Arena.

For private data, a PGDATA structure is imbedded in each process' [PTDA](#) at (**ptda\_pgdata** +0x80).

For the Global Shared Region the PGDATA structure is located at public symbol **\_pgShrData**.

For the System Arena the PGDATA structure is located at public symbol **\_pgData**.

The tables of PTE counts pointed to by **pd\_pcalloc**, **pd\_pcpresent** and **pd\_pcresident** are located as follows:

System arena:

<b>pd_pcalloc</b>	Located at public symbol <b>_pgcPteAllocated</b>
<b>pd_pcpresent</b>	Located at public symbol <b>_pgcPtePresent</b>
<b>pd_pcresident</b>	Located at public symbol <b>_pgcPteResident</b>

Global System Region:

<b>pd_pcalloc</b>	Located at public symbol <b>_pgcShrPteAllocated</b>
<b>pd_pcpresent</b>	Located at public symbol <b>_pgcShrPtePresent</b>
<b>pd_pcresident</b>	Located at public symbol <b>_pgcShrPteResident</b>

Private Data:

All three tables are imbedded contiguously within the **PTDA** at **ptda\_pgc**.

**Note:** The tables pointed to by **pd\_ppde**, **pd\_ppte**, **pd\_pcalloc**, **pd\_pcpresent** and **pd\_pcresident** are indexed relative to the base virtual page number (**pd\_base**) of the region.

**VM Owners.**

**ptda (0xffcb) os2krnl (0xffaa)**

[Format](#)

**PGDATA** Per Arena Page Table Data.

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pd_ppte	+0	4	D	pointer to ptes
pd_ppde	+4	4	D	pointer to pdes
pd_pcalloc	+8	4	D	pointer to (word) counts of allocated ptes per pde
pd_pcpresent	+c	4	D	pointer to (word) counts of present ptes per pde
pd_pcresident	+10	4	D	pointer to (word) counts of locked/resident ptes
pd_base	+14	4	D	base virtual page number
pd_pvdmbalias	+18	4	D	base of vdm alias region
pd_maxpde	+1c	2	W	max potential pdes for this arena
pd_cpdelow	+1e	2	W	count of low in-use pdes
pd_cpdehigh	+20	2	W	count of high in-use pdes
pd_ptcontext	+22	2	W	page table context
pd_flags	+24	2	W	per-process page manager flags

#### pd\_flags flag definitions:

Name	Bit	Mask	Description
PD_AGINGNEEDED	1		process not aged in this sweep
PD_FREE	2		

-----

## Per Arena Page Table Data for OS/2 Warp 3.0 SMP

#### PGDATA Per Arena Page Table Data.

<i>Field Name</i>	<i>Off</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pd_ppte	+0	4	D	pointer to ptes
pd_ppde	+4	4	D	pointer to pdes
pd_pcalloc	+8	4	D	pointer to (word) counts of allocated ptes per pde
pd_pcpresent	+c	4	D	pointer to (word) counts of present ptes per pde
pd_pcresident	+10	4	D	pointer to (word) counts of locked/resident ptes
pd_base	+14	4	D	base virtual page number
pd_pvdmbalias	+18	4	D	base of vdm alias region
pd_cpdelo	+1c	2	W	count of low in-use pdes
pd_cpdehi	+1e	2	W	count of high in-use pdes

pd_maxpde	+20	2	W	max potential pdes for this arena
pd_ptcontext	+22	2	W	page table context
pd_flags	+24	2	W	per-process page manager flags

-----

## VM Arena Header

### Locations

**\_ahvmSys** locates the System Arena VMAH.

**\_ahvmShr** locates the Shared Arena VMAH.

PTDA field **ptda\_ah** locates each Private Arena VMAH.

### VM Owner

For shared and system arenas: **os2krnl (0xffaa)**

For private arenas: **ptda (0xffcb)**

### Format

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ah_pahNext	+0	4	D	Link to next arena
ah_pahPrev	+4	4	D	Link to previous arena
ah_parSen	+8	4	D	Handle of arena sentinel
ah_parFree	+c	4	D	Hint of 1st free block in arena
ah_papbm	+10	4	D	Pointer to bitmap directory
ah_paharHash	+14	4	D	Hash table pointer
ah_pat	+18	4	D	Pointer to per-type info
ah_fl	+1c	4	D	Flags
ah_laddrMin	+20	4	D	Minimum address currently mapped
ah_laddrMax	+24	4	D	Max address currently mapped
ah_car	+28	4	D	Count of arena entries
ah_carBitmap	+2c	4	D	Max entry count to need bitmap
ah_lbmNumbMax	+30	4	D	Max bitmap number
ah_lbmeNumbMax	+34	4	D	Max bitmap entry number
ah_lHashNumbMax	+38	4	D	Max hash table index
ah_hob	+3c	2	W	Arena header pseudo-handle
ah_filler	+3e	2	W	Make structure 4-byte multiple

### ah\_fl flag definitions:

Name	Bit Mask	Description
VMAH_BITMAP_BYPASS	0x00000001	Worth bypassing bitmap

VMAH\_NO\_HASH\_WRAP 0x00000002 No hash table wraparound yet  
VMAH\_GROW\_DOWN 0x00000004 Arena grows down

-----

## VM Alias Record

### Pointers

`_paVMAliases` points to the VMAL table.

### VM Owner

`vmal (0xffe2)`

### Format

Field Name	Offset	Length	Type	Description
<code>vmal</code>	+0	8	S	VM alias record
<code>al_har</code>	+0	2	W	handle to alias' arena record
<code>al_hobptda</code>	+2	2	W	context the alias is created from
<code>al_pgoff</code>	+4	2.4	D	page offset of the alias from start of object
<code>al_f</code>		1.4		flags indicating type of alias
<code>vmsal</code>	+0	8	S	SEL alias record
<code>sal_har</code>	+0	2	W	handle to alias' arena record
<code>sal_selcode</code>	+2	2	W	code selector if cs alias
<code>al_hobptda</code>	+2	2	W	context the alias is created from if MEMMAP alias
<code>sal_cref</code>	+4	1.2	D	reference count
<code>sal_f</code>		0.6		flags
<code>sal_seldata</code>	+6	2	W	data selector if cs alias (unused for MEMMAP)

### `al_f` flag definitions:

Name	Bit Mask	Description
<code>AL_ISBUSY</code>	0x1	Set if record is busy
<code>AL_CSALIAS</code>	0x2	Set if cs alias record
<code>AL_MEMMAP</code>	0x4	Set if MemMapAlias record
<code>AL_DBGALIAS</code>	0x8	Set if debug alias
<code>AL_CSDSVALID</code>	0x10	Set if ds selector valid
<code>AL_DEVHLP</code>	0x20	Set if Devhlp alias
<code>AL_PRIV</code>	0x40	Set if privatized alias

AL_VDM	0x80	Set if VDM alias
AL_NOALIAS	0x100	Set if UVIRT mapping in VDMs

#### sal\_f flag definitions:

Name	Bit Mask	Description
SAL_CSALIAS	AL_CSALIAS	
SAL_MEMMAPALIAS	AL_MEMMAP	
SAL_CSDSVALID	0x10	mustn't coincide with other alias types
SAL_ALIASREFSHIFT	0x6	Low six bits reserved for flags
SAL_ALIASREFMASK	0x0ffc0	reference count bits mask

-----

## VM Arena Record

#### Pointers

`_parvmOne` points to the VMAR table.

#### VM Owner

`vmar (0xfe3)`

#### Format

Field Name	Offset	Length	Type	Description
vmar_reg	+0	16	S	Regular Arena Record
ar_xf	+0	1.4	D	Extra flags
ar_cpg		2.4		Size in pages
ar_ipg	+4	2.4	D	Virtual page no.
ar_f		1.4		Flags
ar_harnext	+8	2	W	Handle of next Arena Record
ar_harprev	+a	2	W	Handle of previous Arena Record
ar_harlink	+c	2	W	Handle of associated Arena Record
ar_harhash	+e	2	W	Hash table link
ar_hob	+10	2	W	Handle of Object Record
ar_hco	+12	2	W	Context record handle (shar+shr data)
ar_hobptda	+12	2	W	PTDA handle or NULL (prvar or shar + instance data)
ar_sel	+12	2	W	Selector (sysarena only)
ar_hal	+14	2	W	Alias record handle, * =0 means not an alias

vmar_sen	+0	16	S	Sentinel Arena Record
ar_xf	+0	1.4	D	Extra flags
ar_cpg	+1.4	2.4		Size in pages
ar_ipg	+4	2.4	D	Virtual page no.
ar_f	+6.4	1.4		Flags
ar_harnext	+8	2	W	Handle of next Arena Record
ar_harprev	+a	2	W	Handle of previous Arena Record
ar_harlink	+c	2	W	Handle of associated Arena Record
ar_harhash	+e	2	W	Hash table link
ar_ipgmax	+10	4	D	Maximum lage no. in the arena
ar_unused	+14	2	W	reserved

#### ar\_f flag definitions:

Name	Bit Mask	Description
AR_INUSE	0x001	Record not on free list
AR_TAG	0x006	Record type mask
AR_TAGREG	0x000	Regular record
AR_TAGSEN	0x002	Sentinel
AR_TAGBSEN	0x006	Boundary sentinel
AR_SELMAP	0x008	Memory mapped by selector
AR_SELBASEALL	0x00c	Base selector map all
AR_SELMASK	0x00c	Selector map mask
AR_RELOAD	0x010	Pre-reserved for huge item or
AR_WRITE	0x020	Write permission
AR_USER	0x040	User pages
AR_EXEC	0x080	Executable Pages
AR_READ	0x100	Read permission
AR_HCO	0x200	Record linked to Context List
AR_GUARD	0x400	Guard pages
AR_SGS	0x800	Registered under Screen Group Switch

#### ar\_xf flag definitions:

Name	Bit Mask	Description
AR_HCOH	0x001	context record handle > 64k

-----

## VM Arena Type Information Record

## Pointers

VMAH field **ah\_pat** points to the associated VMAT.

## Locations

**\_atvm** locates the table of VMATs.

## VM Owner

**os2krnl (0xffaa)**

## Format

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
at_laddrInitMin	+0	4	D	Initial minimum
at_laddrInitMax	+4	4	D	Initial maximum
at_laddrAbsMin	+8	4	D	Abs minimum boundary
at_laddrAbsMax	+c	4	D	Abs minimum boundary
at_cbInitBetween	+10	4	D	Spacer between arenas
at_lHashNumbMask	+14	4	D	Hash number mask
at_lHashNumbShift	+18	4	D	Hash number shift
at_lHashNumbAbsMax	+1c	4	D	Max hash table index
at_lHashMinSize	+20	4	D	Min hash table size
at_lbmNumbMask	+24	4	D	Bitmap number mask
at_lbmNumbShift	+28	4	D	Bitmap number shift
at_lbmNumbAbsMax	+2c	4	D	Abs Max bitmap #
at_lbdMinSize	+30	4	D	Min bitmap dir size
at_lbmeNumbMask	+34	4	D	Bitmap entry # mask
at_lbmeNumbShift	+38	4	D	Bitmap entry # shift
at_lbmeNumbAbsMax	+3c	4	D	Abs Max bitmap entry
at_lbmeBitNumbMask	+40	4	D	Bit number mask
at_lbmeBitNumbShift	+44	4	D	Bit number shift
at_flInit	+48	4	D	Initial flags
at_lGran	+4c	4	D	Granularity
at_laddrMinNoWrap	+50	4	D	Min no-hash wrap laddr
at_laddrMaxNoWrap	+54	4	D	Max no-hash wrap laddr
at_harParent	+58	2	W	Parent arena

## at\_flInit flag definitions:

Name	Bit	Mask	Description
VMAT_PRIV_TILED	0		
VMAT_PRIV_VDM	1		
VMAT_SHR_TILED	2		
VMAT_SYS	3		



VMAT\_MAX

VMAT\_SYS

-----

## VM Context Record

### Pointers

`_pcovmOne` points to the table of VMCOs.

### Locations

System Arena.

### VM Owner

`vmco (0xffe5)`

### Format

Field Name	Offset	Length	Type	Description
<code>co_hconext</code>	+0	2	W	Index of next Context Record
<code>co_hobptda</code>	+2	2	W	PTDA handle
<code>co_fb</code>	+4	1	B	Context record flags

### `co_fb` flag definitions:

Name	Bit Mask	Description
<code>CO_CREATOR</code>	<code>0x01</code>	originating context
<code>CO_PRIV</code>	<code>0x80</code>	Privatized context
<code>CO_HCOH</code>	<code>0x20</code>	Next context record handle > 64k
<code>CO_WRITE</code>	<code>0x02</code>	Write permission
<code>CO_USER</code>	<code>0x04</code>	User storage
<code>CO_EXEC</code>	<code>0x08</code>	Executable
<code>CO_READ</code>	<code>0x10</code>	Read permission
<code>CO_GUARD</code>	<code>0x40</code>	Guard page

-----

## VM Object Record

### Pointers

`_pobvmOne` points to the table of VMOBs.

### Locations

System Arena.

**VM Owner****vmob (0xffff1)****Format**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ob_har	+0	2	W	Arena Record handle
ob_hobnext	+2	2	W	Associated Object Record handle
ob_va	+0	4	D	Pseudo-object's virtual address
ob_fs	+4	2	W	Flags
ob_hobowner	+6	2	W	Owner i.d.
ob_hobmte	+8	2	W	MTE handle
ob_wsemowner	+a	2	W	I.d. of thread owning semaphore
ob_bsemcnt	+c	1	B	Counter and waiting flag
ob_cllock	+d	1	B	Count of all long-term locks
ob_cslock	+e	1	B	Count of all short-term locks
ob_xflags	+f	1	B	Extra flags

**Note:**

A complete list of system owner ids may be found under [VM System Object Owner Ids](#) in the [Reference Tables](#) section of the System Reference.

**ob\_fs flag definitions:**

Name	Bit Mask	Description
OB_PSEUDO	0x8000	Pseudo-object
OB_API	0x4000	API allocated object
OB_LOCKWAIT	0x2000	Some thread to wake in VMUnlock
OB_LALIAS	0x1000	Object has aliases
OB_SHARED	0x0800	Object's contents are shared
OB_UVIRT	0x0400	UVirt object
OB_ZEROINIT	0x0200	Object is zero-initialized
OB_RESIDENT	0x0100	Initial allocation was resident
OB_LOWMEM	0x0040	Object is in low memory
OB_GUARD	0x0080	Page attribute/permission flags
OB_EXEC	0x0020	Executable
OB_READ	0x0010	Read permission
OB_USER	0x0008	User Storage
OB_WRITE	0x0004	Write permission
OB_HUGE	0x0002	Object is huge
OB_SHRINKABLE	0x0001	Object is Shrinkable

OB\_DHSETMEM    0x0001    DevHlp\_VMSetMems are allowed

#### ob\_xflags flag definitions:

Name	Bit Mask	Description
VMOB_SLOCK_WAIT	0x01	Waiting on short term locks to clear
VMOB_LLOCK_WAIT	0x02	Waiting on long term locks to clear
VMOB_DISC_SEG	0x04	Object is part of a discardable seg
VMOB_HIGHMEM	0x08	Object was allocated via dh_vmalloc

---

## Virtual Page Structure

#### Pointers

**pf\_pvp** points to the head of the VP array.

#### Locations

System Arena.

#### VM Owner

**pgvp (0xffb3)**

#### Format

Field Name	Off	Length	Type	Description
avp_s	+0	c	S	active vp
vp_frame	+0	2.4	D	frame, swp or ldr block #
vp_flags		1.4		flags
vp_obpg	+4	2	W	object relative page number
vp_hob	+6	2	W	handle to object record
vp_refcount	+8	2	W	virtual page reference count
vp_semowner	+a	2	W	Slot number of semaphore owner
fvp_s	+0	a	S	Free vp
vp_flink	+0	4	D	forward link
vp_blink	+4	4	D	backward link
	+8	2	W	pad

#### vp\_flag flag definitions:

name	bit mask	description
VP_BUSY	0x001	page semaphore taken
VP_WANTED	0x002	page semaphore requested

VP_CACHE	0x004	search page cache for pf
VP_PFIDLE	0x008	cross linked to idle pf
VP_PF	0x010	cross linked to pf
VP_DF	0x020	has swap file disk frame
VP_DIRTY	0x040	contents written to - from pte
VP_SHDIRTY	0x080	shadow dirty bit (for VDMs)
VP_SOW	0x100	change to swappable on write
VP_PRIVATIZED	0x200	vp privatized
VP_RESIDENT	0x400	cannot be moved - value from pte
VP_DISCARDABLE	0x800	1 = discardable, 0 = swappable

-----

## KernelHeap Header Structure

### Pointers

**\_apkh** points to the head of the VMKH array.

**SAS\_vm\_heap\_info** also points to this array.

### Locations

System Arena.

### VM Owner

**os2krnl (0xffaa)**

### Format

**VMKH** Kernel Heap Header.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
kh_fl	+0	4	D	Allocation flags
kh_pkrh	+4	4	D	Pointer to resident heap header
kh_pksh	+4	4	D	Pointer to swappable heap header

**kh\_fl** flag definitions.

Name	Bit Mask	Description
PG_CONTIG	0x00000001	contiguous physical memory
PG_NOINCR	0x00000001	don't increment physical addrs
PG_W	0x00000002	Writable - value from pte
PG_U	0x00000004	user mode accessible - from pte
PG_X	0x00000008	eXecutable

PG_R	0x00000010	Readable
PG_1M	0x00000020	must reside below 1 meg physical
PG_GUARD	0x00000040	guard page - from pte
PG_16M	0x00000040	must reside below 16 meg physical
PG_ZEROFILL	0x00000080	zero initialize pages
PG_SWAPONWRITE	0x00000100	value from vp
PG_UVIRT	0x00000200	value from pte
PG_RESIDENT	0x00000400	value from pte
PG_DISCARDABLE	0x00000800	value from vp

### Heap Handles (hkh)

Kernel heap handles used to index the array of VMKH structures.

Name	hkh value	Description
VM_HKH_PUB_RESRW	1	Kernel resident RW heap handle
VM_HKH_PUB_RESRO	2	Public resident RO heap handle
VM_HKH_PUB_SWAPRW	3	Public swappable RW heap handle
VM_HKH_PUB_SWAPRO	4	Public swappable RO heap handle
VM_HKH_PUB_RES1MRW	5	Public resident RW 1M handle
VM_HKH_PUB_RES1MRO	6	Public resident RO 1M handle

### Note:

It is possible for more than one handle to be served by the same heap. In particular, under the **RETAIL** kernel all heap handle are mapped to either a read/write resident or swappable heap.

## Kernel Resident Heap Structures

### Pointers

For resident heap entries, **kh\_pkrh** of a VMKH entry points to a VMKRH.

An array of VMKRHY structures are embedded in VMKRH at **krh\_akrhy**.

An array of VMKRHS structures are embedded in VMKRH at **krh\_akrhs**.

**krhf\_pbNext** and **krhb\_pbPrev** double link VMKRHF structures from the dummy VMKRHF embedded in VMKRK at **krh\_krhfDummy**

### Locations

System Arena.

VMKRH prefixes a resident heap.

Allocated blocks (VMKRHB and VMKRHBA) are sparsely allocated from the heap, with interstitial free blocks (VMKRHF).

VMKRHB prefixes the data portion of an allocated block.

VMKRHBA suffixes the data portion of an allocated block when the **krhb\_attr** bit is set in the associated VMKRHB.

#### VM Owners

**vmkrhro (0xffeb)**

**vmkrhrw (0xffec)**

**kdbsym (0xff7c)**

**krhrw1m (0xff43)**

**krhro1m (0xff44)**

#### Format

**VMKRH** Kernel Resident Heap Header.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
krh_cBlocks	+0	4	D	Count of heap blocks
krh_cFreeBlocks	+4	4	D	Count of free heap blocks
krh_pbFirst	+8	4	D	First Heap Block
krh_pbLast	+C	4	D	Last Heap Block
krh_pbEndRes	+10	4	D	Upper bound of reserved virt mem
krh_pbEndCom	+14	4	D	Upper bound of committed virt mem
krh_lpgBase	+18	4	D	Start lin page of heap object
krh_cMods	+1C	4	D	Count of heap modifications
krh_fl	+20	4	D	Resident heap flags
krh_krhfDummy	+24	C	D	Dummy free block
krh_pkrhsLast	+30	4	D	Last Freelist Section Pointer
krh_akrhy	+34	30	S	Array of 8 yield structures
krh_akrhs	+64	50	S	Array of 5 free list sections
krh_hob	+B4	2	W	Heap object handle
krh_ksem	+B6	C(10)	S	KSEM for resident heap

**VMKRHY** Kernel Resident Heap Yield List Structure.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
krhy_pb	+0	4	D	Block pointer
krhy_cyield	+4	2	W	Yield count

**VMKRHS** Kernel Resident Heap Free List Anchor Structure.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
krhs_cbMax	+0	4	D	Maximum block size
krhs_krhfHead	+4	C	S	Dummy free block used to locate head of list

#### VMKRHF Kernel Resident Heap Free Block.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
krhf_struct0	+0	4	S	Resident Heap Block Header
krhf_pbNext	+4	4	D	Forward freelist link
krhf_pbPrev	+8	4	D	Backward freelist link

#### VMKRHB Kernel Resident Heap Block Header.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
	+0	4	S	Regular Block Header
krhb_hobowner	+0	2	D	Owner
krhb_pfree		0.1		Preceding block is free
krhb_usSize		1.5		Size of block in dwords
krhb_yield		0.1		Thread-yielded-here flag
krhb_attr		0.1		Attributed block flag (=0)
	+0	4	S	Attributed Block Header
	+0	0.1	D	Attributed block is free
		0.1		Block preceding attributed block is free
krhb_ulSize		3.4		Size of block in dwords
krhb_yield		0.1		Thread-yielded-here flag
krhb_attr		0.1		Attributed block flag (=1)

#### VMKRHBA Kernel Resident Heap Block Header Attributes.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
khba_sel	+0	2	W	Selector mapping heap block data
khba_hobOwner	+2	2	W	Heap block owner
khba_hobMTE	+4	2	W	Heap block MTE
khba_pad	+6	2	W	Pad to DWORD mutiple.

#### krh\_fl flag definitions.

<i>Name</i>	<i>Bit Mask</i>	<i>Description</i>
PG_CONTIG	0x00000001	contiguous physical memory
PG_NOINCR	0x00000001	don't increment physical addrs
PG_W	0x00000002	Writable - value from pte
PG_U	0x00000004	user mode accessible - from pte
PG_X	0x00000008	eXecutable
PG_R	0x00000010	Readable

PG_1M	0x00000020 must reside below 1 meg physical
PG_GUARD	0x00000040 guard page - from pte
PG_16M	0x00000040 must reside below 16 meg physical
PG_ZEROFILL	0x00000080 zero initialize pages
PG_SWAPONWRITE	0x00000100 value from vp
PG_UVIRT	0x00000200 value from pte
PG_RESIDENT	0x00000400 value from pte
PG_DISCARDABLE	0x00000800 value from vp

-----

## Kernel Swappable Heap Structures

### Pointers

For swappable heap entries, **kh\_pksh** of a VMKH entry points to a VMKSH.

An array of VMKRHY structures are embedded in VMKRH at **krh\_akrhy**.

An array of VMKRHS structures are embedded in VMKRH at **krh\_akrhs**.

**ksh\_hdrEntry** in VMKSH points to the first chained VMKSHD.

**kshd\_pbNext** in VMKSHD points to subsequent VMKSHD structures. VMKRHF structures from the dummy VMKRHF embedded in VMKRK at **krh\_krhfDummy**

### Locations

System Arena.

VMKSHD structure are allocated from the Kernel Resident Heap.

VMKSH prefixes a swappable heap.

Allocated blocks (VMKSHB) are sparsely allocated from the heap.

VMKSHB prefixes the data portion of an allocated block.

### VM Owners

VMKSH and VMKSHB:

**vmkshro (0xffee)**

**vmkshrw (0xffef).**

VMKSHD:

**vmshd (0xffed).**

### Format

**VMKSH** Kernel Swappable Heap Header (RETAIL kernel)

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ksh_hdrEntry	+0	12	S	Dummy head descriptor of free chain
	+0	4	D	



	+4	4	D	pointer to next descriptor
	+8	4	D	
ksh_pHint	+c	4	D	Descriptor pointer to last block touched
ksh_pHPrev	+10	4	D	Descriptor pointer to PrevBlk of last touched
ksh_HintSize	+14	4	D	Size of last block touched
ksh_pbEndRes	+18	4	D	Upper bound of reserved virt mem
ksh_pbEndCom	+1c	4	D	Upper bound of committed virt mem
ksh_fl	+20	4	D	Swappable heap allocation flags
ksh_hob	+24	2	W	Heap object handle
ksh_pbStart	+26	4	D	Lower bound of reserved virt mem
ksh_ksem	+2a	C	S	KSEM for swappable heap

#### VMKSH Kernel Swappable Heap Header (**ALLSTRICT** kernel)

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ksh_hdrEntry	+0	4	D	Head of descriptor chain
ksh_pHint	+4	4	D	Descriptor pointer to last block touched
ksh_pHPrev	+8	4	D	Descriptor pointer to PrevBlk of last touched
ksh_HintSize	+C	4	D	Size of last block touched
ksh_pbEndRes	+10	4	D	Upper bound of reserved virt mem
ksh_pbEndCom	+14	4	D	Upper bound of committed virt mem
ksh_fl	+18	4	D	Swappable heap allocation flags
ksh_hob	+1C	2	W	Heap object handle
ksh_pbStart	+1E	4	D	Lower bound of reserved virt mem
ksh_ksem	+22	10	S	KSEM for swappable heap
ksh_ckshds	+32	4	D	Count of heap modifications
ksh_cFrees	+36	4	D	Count of heap modifications
ksh_cAllocs	+3a	4	D	Count of heap modifications

#### VMKSHD Kernel Swappable Heap Descriptor Record for Free Blocks.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
kshb_size	+0	4	D	Size of block in dwords
kshd_pNext	+4	4	D	Next free block
kshd_pb	+8	4	D	Address of block header

#### VMKSHB Kernel Swappable Heap Block Header for Allocated blocks.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	---------------	---------------	-------------	--------------------

	+0	1	D	Signature 0x52
kshb_size		3		size of block
kshb_hobowner	+4	2	W	Owner
kshb_sel	+6	2	W	Selector

**ksh\_fl** flag definitions.

Name	Bit Mask	Description
PG_CONTIG	0x00000001	contiguous physical memory
PG_NOINCR	0x00000001	don't increment physical addrs
PG_W	0x00000002	Writable - value from pte
PG_U	0x00000004	user mode accessible - from pte
PG_X	0x00000008	eXecutable
PG_R	0x00000010	Readable
PG_1M	0x00000020	must reside below 1 meg physical
PG_GUARD	0x00000040	guard page - from pte
PG_16M	0x00000040	must reside below 16 meg physical
PG_ZEROFILL	0x00000080	zero initialize pages
PG_SWAPONWRITE	0x00000100	value from vp
PG_UVIRT	0x00000200	value from pte
PG_RESIDENT	0x00000400	value from pte
PG_DISCARDABLE	0x00000800	value from vp

-----

## Scheduler Thread and Process Control Block Reference

The following control blocks are described in this section:

[Thread Control Block \(TCB\)](#)

[Thread Swappable Data \(TSD\)](#)

[Per Task Data Area \(PTDA\)](#)

[The local exception handler Long-Jump Buffer \(ljmp\)](#)

[Local Information Segment \(LISEG\)](#)

[Global Information Segment \(GISEG\)](#)

[Process Information Block \(PIB\)](#)

[Thread Information Block \(TIB\)](#)

[System Stack Frames and Client Register Information](#)

[Exit List Data Structure \(EXENT\)](#)

[Exception Handler Structures](#)

An overview of the Scheduler Control Blocks follows:

---

## Scheduler and Task Management Control Block Diagrams

The following diagrams illustrate the relationships between various Scheduler and Task Management control blocks:

[Process Management](#)

[Thread Management](#)

[Scheduler Finite State Machine](#)

[Thread Tree for a Process](#)

[Process Trees, Subtrees and Zombies](#)

[Orphaned and Adopted Processes](#)

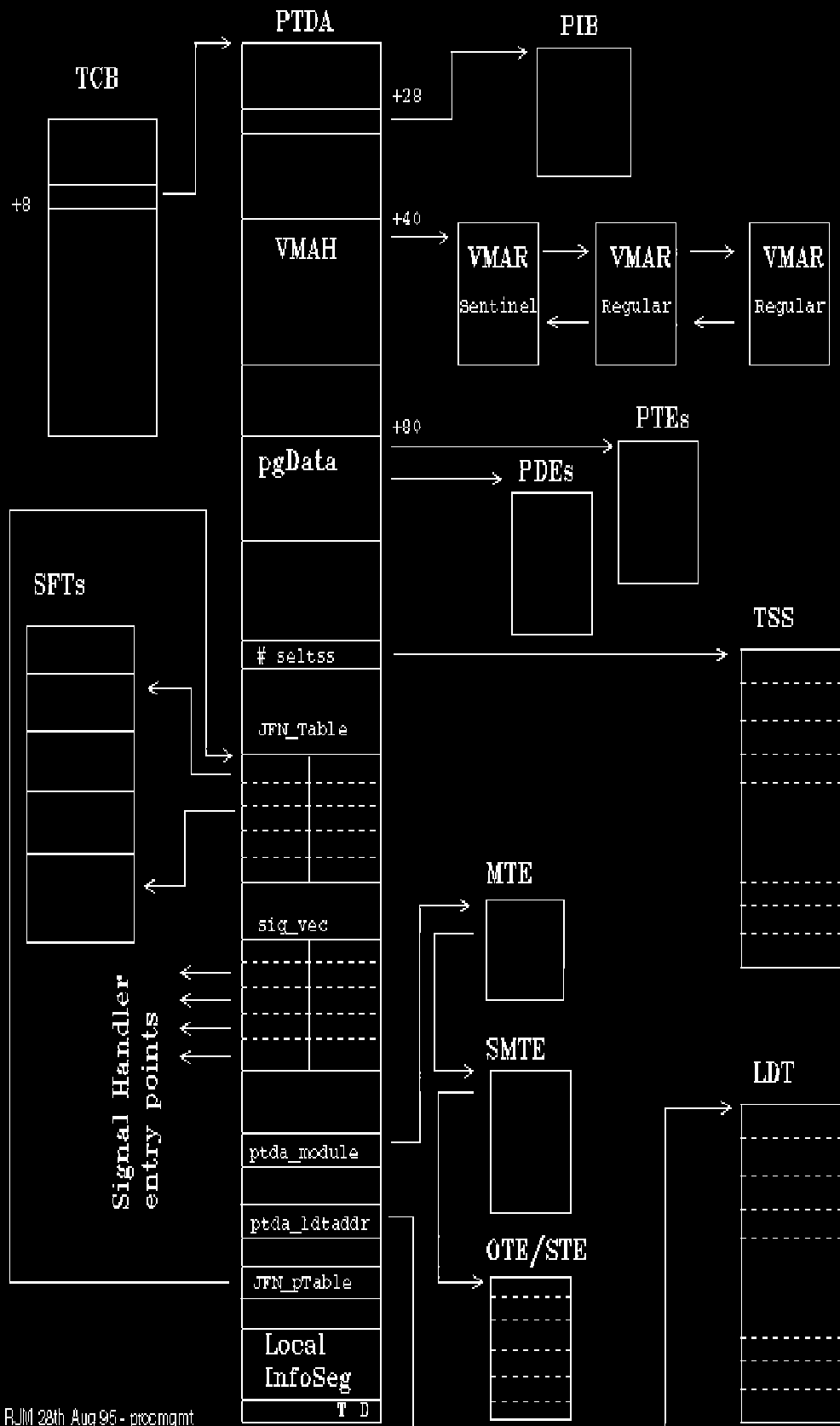
[Exception Management - Overview](#)

[Exception Handler Stack Frames](#)

---

## Process Management

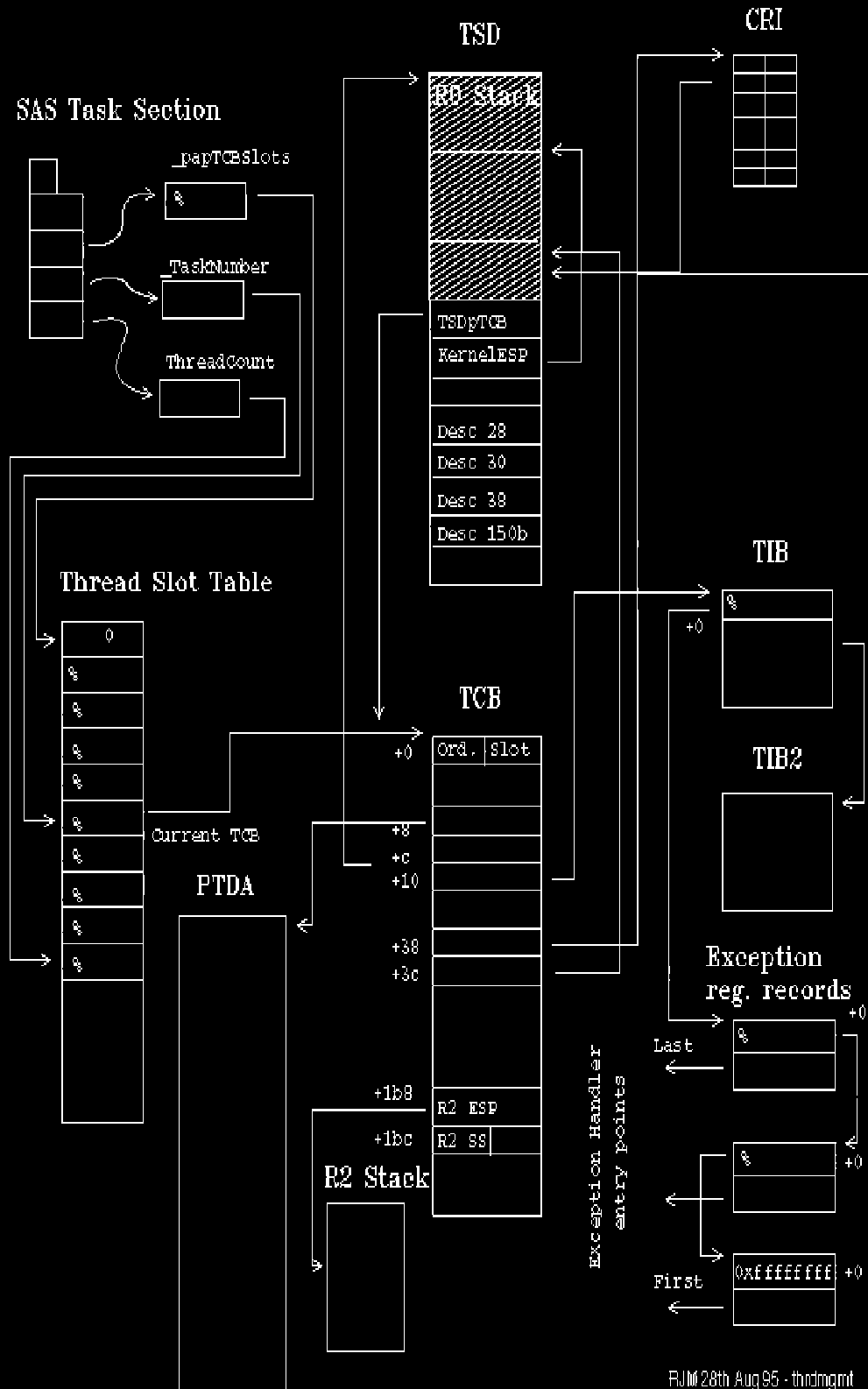
# Process Management



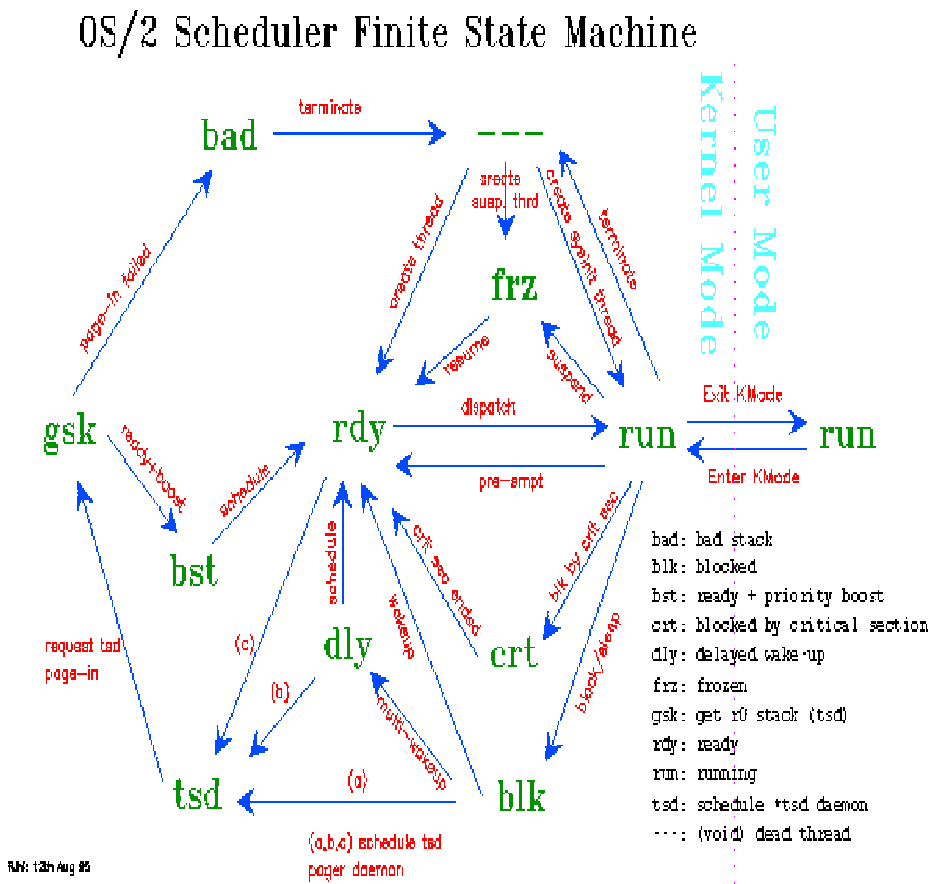
---

## Thread Management

# Thread management

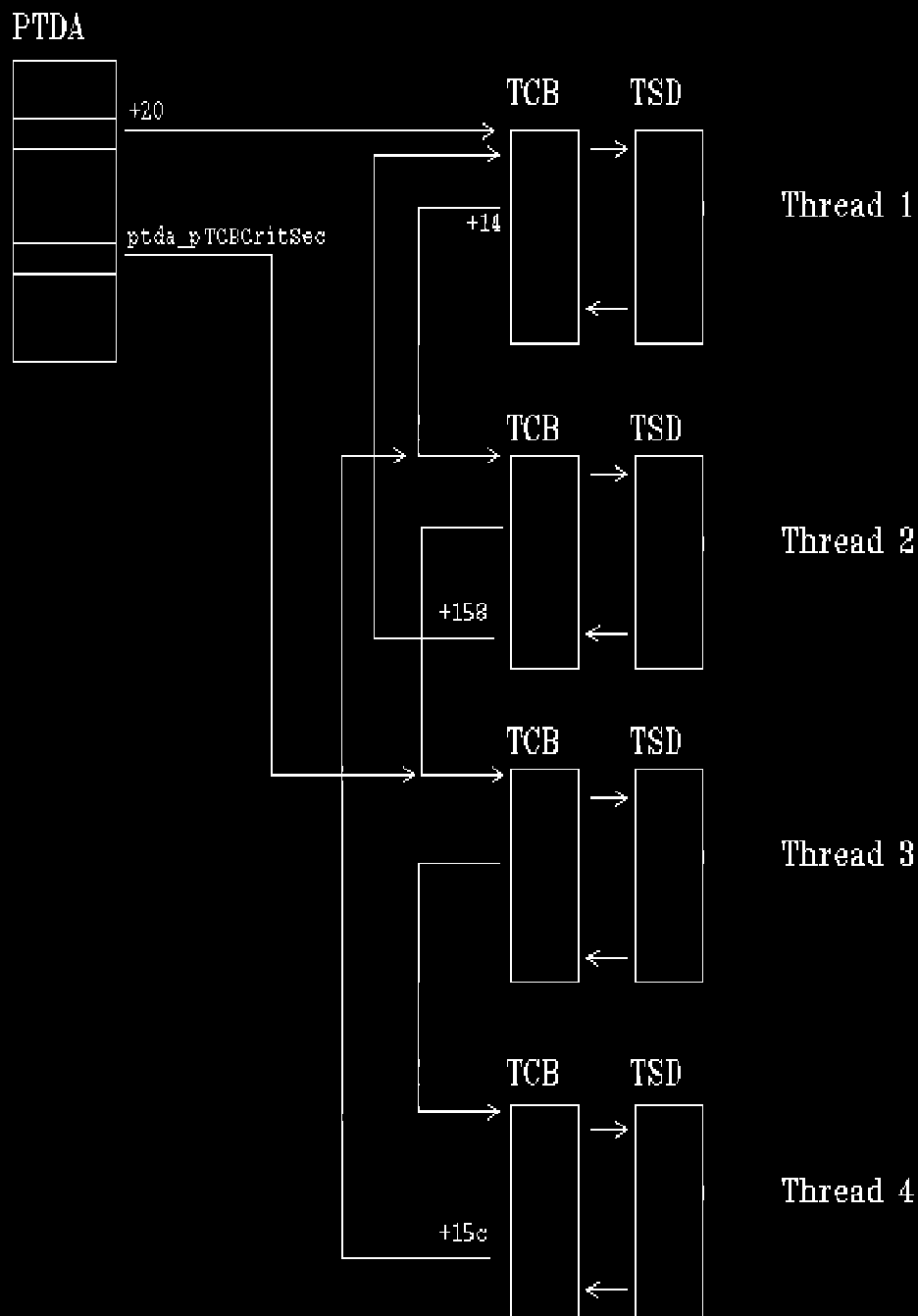


# Scheduler Finite State Machine



# Thread Tree for a Process

## Thread Tree for a Process



Thread 3 is in critical section

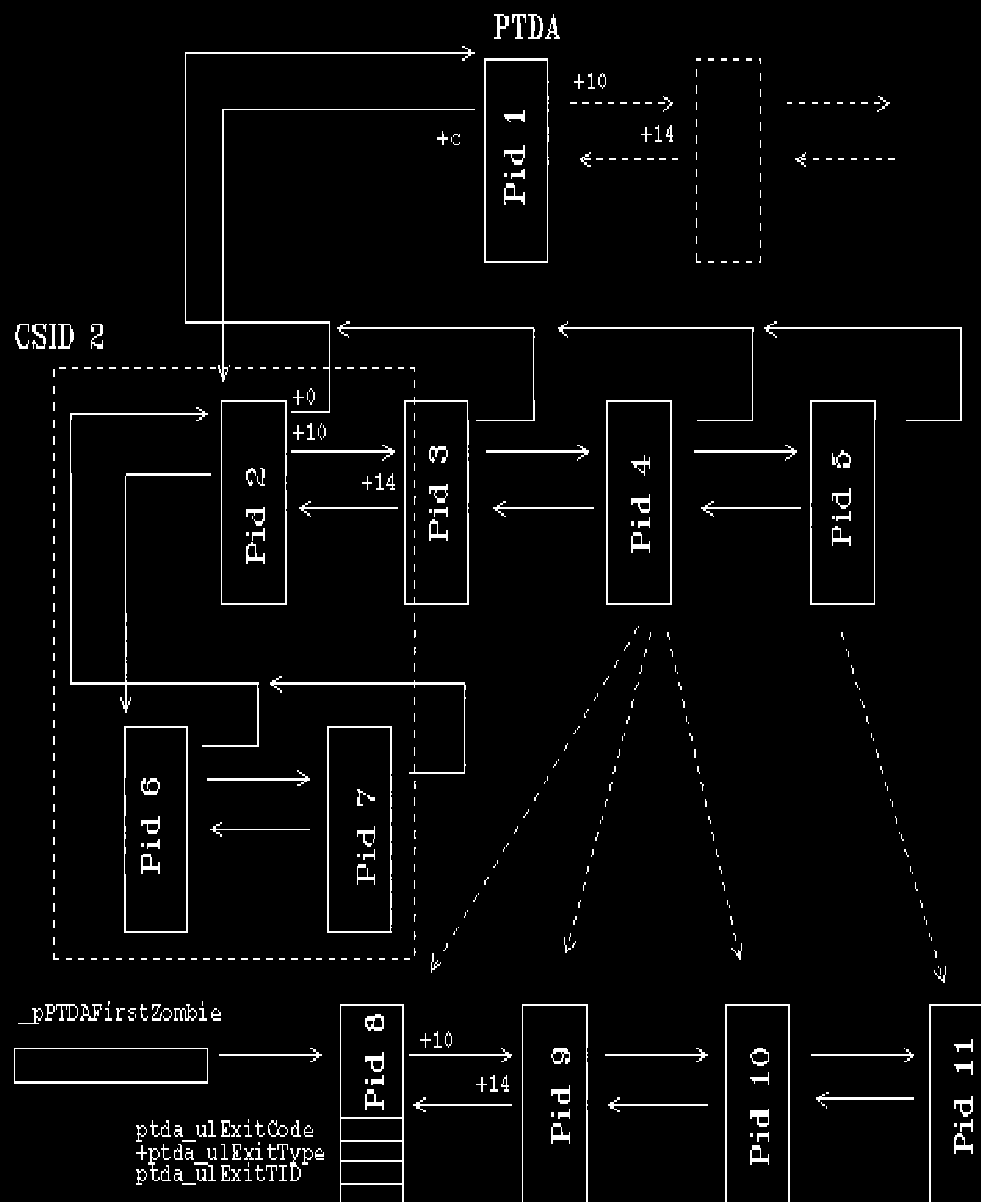
Thread 2 and Thread 1 are waiting for Thread 4 to die



---

## Process Trees, Subtrees and Zombies

# The Process Tree, Subtrees and Zombies



Pid 1 is Detached (no parent)

Other Detached Processes are Siblings of Pid 1

Pids 1 - 7 are active

Pids 8 - 11 are dead (zombies)

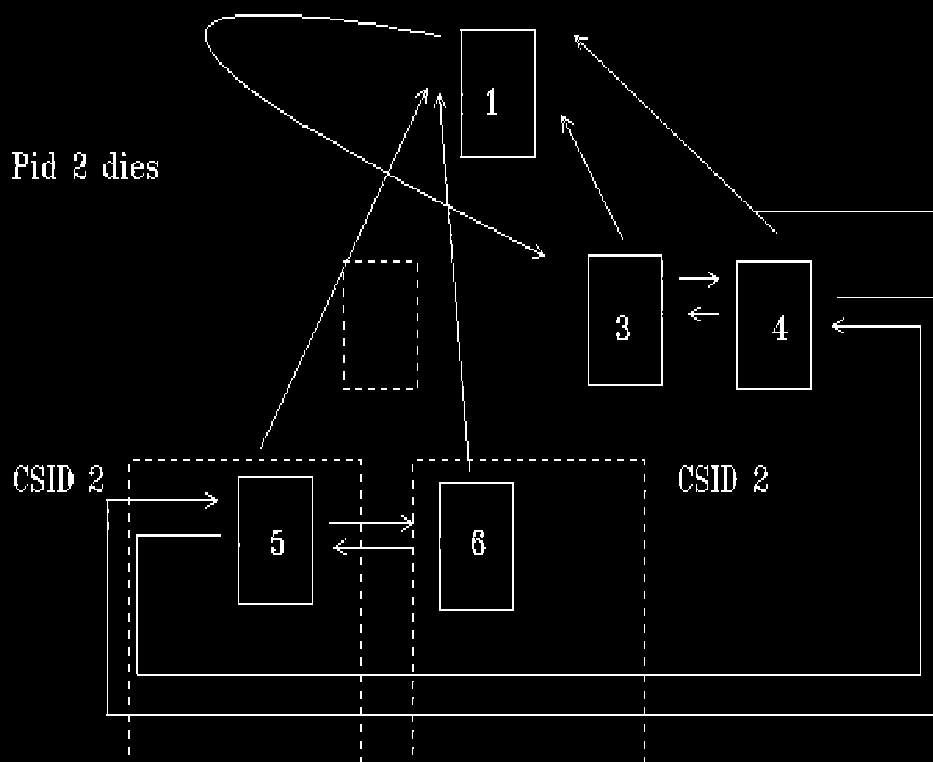
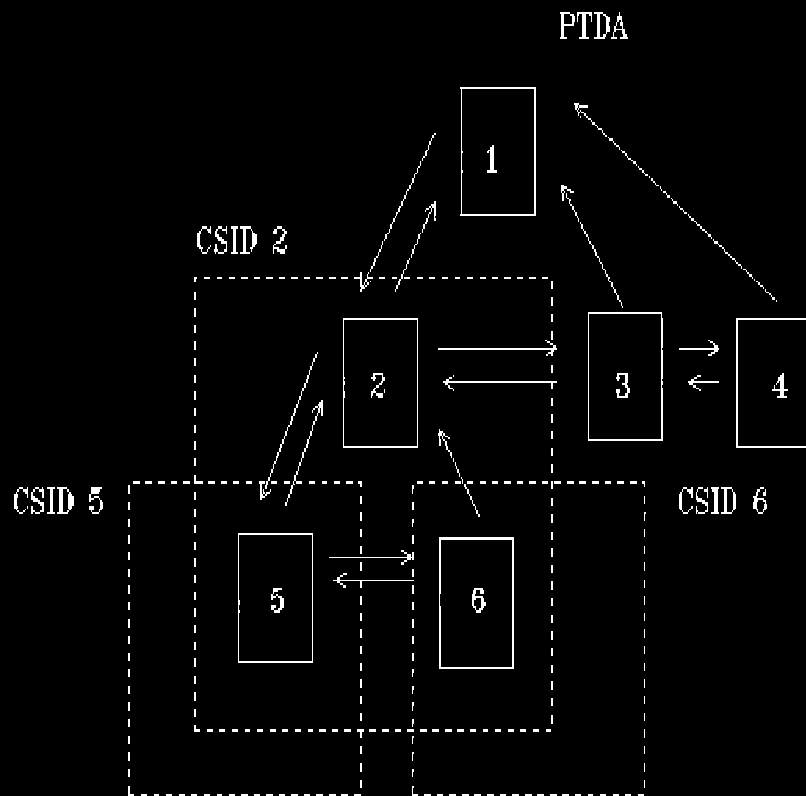
Pid 4 may DosWaitChild on Pids 8 - 10

Pid 5 may DosWaitChild on Pid 11

---

## Orphaned and Adopted Processes

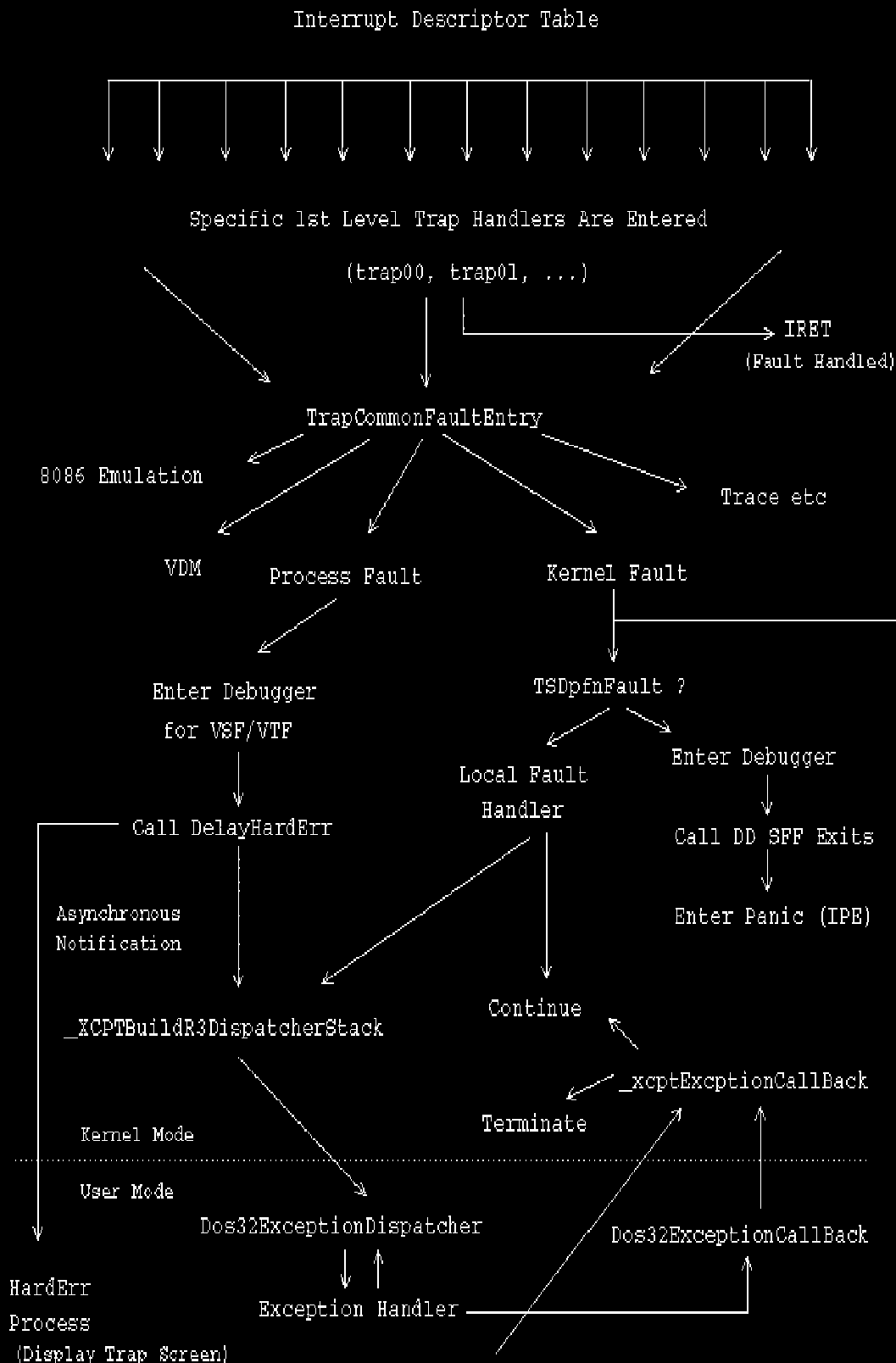
# Orphaned and Adopted Processes



---

## OS/2 Exception Management - Overview

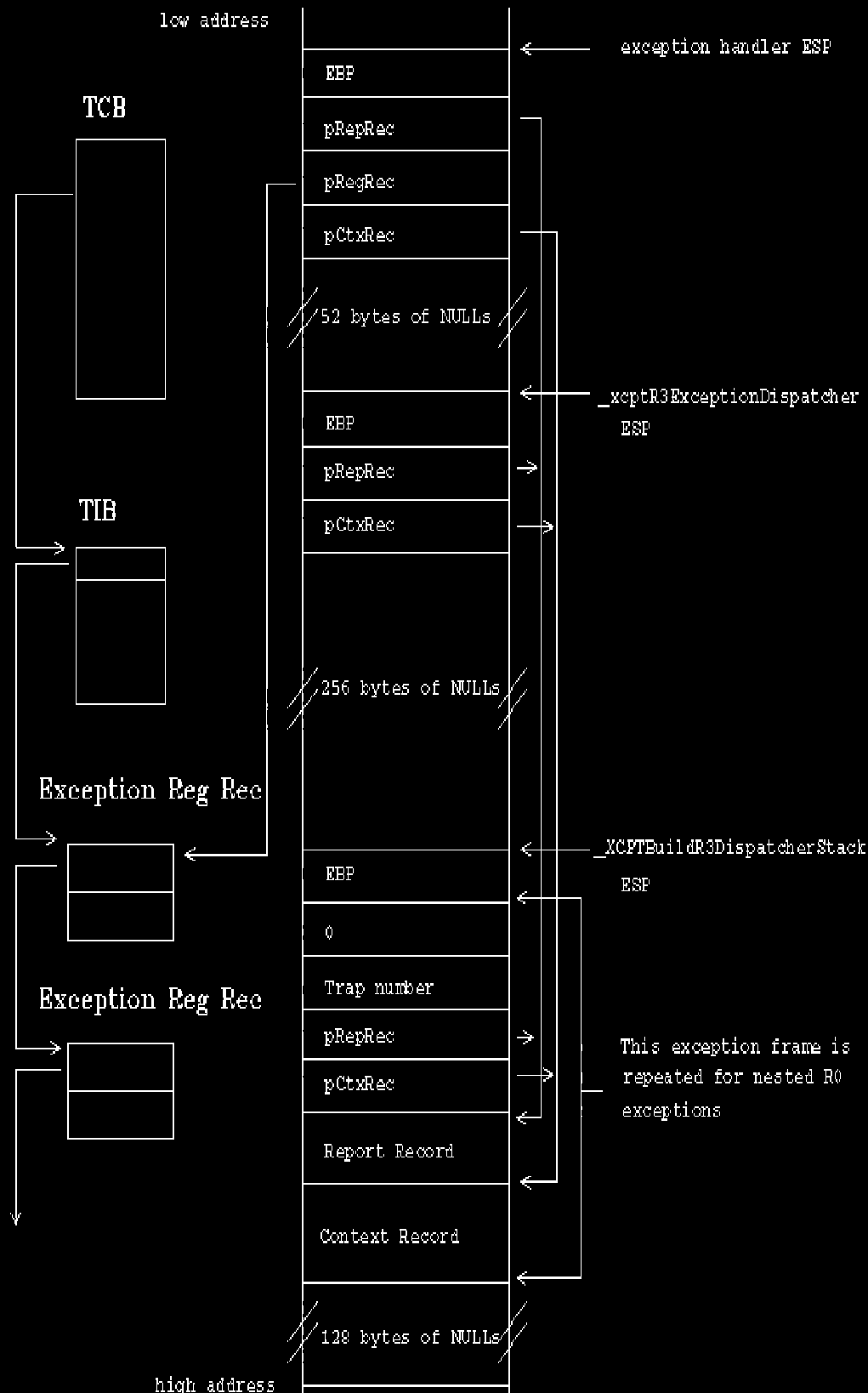
# Exception Handling - Overview



-----

# Exception Handler Stack Frames

# Exception Handler Stack Frames





---

# Thread Control Block OS/2 Warp V4.0

For **TCB** formats for other versions of OS/2 see:

**TCB** for OS/2 Warp V3.0

**TCB** for OS/2 Warp V3.0 with Fix pack 9

**TCB** for OS/2 Warp V3.0 with Fix pack 11 or later

**TCB** for OS/2 V2.11 with Fix pack 90 or later

**TCB** for OS/2 V2.11

## Pointers

**\_papTCBSlots** points to the thread slot table of TCB pointers.

Multiple chain pointers between, TSD, TCB and PTDA.

**CurrTCB** points to the current TCB.

## Locations

System Arena.

## VM Owner

**tcb (0xffcc)**

## Format

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head

TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBpDHRetAddr	+74	4	D	82818 Pointer to DHRouter return address
TCBTLMA	+78	80	D	Thread local memory area
TCBDMAAdd	+f8	4	D	User's I/O transfer address
TCBSecPos	+fc	4	D	Position of first sector accessed within file
TCBThisSFT	+100	4	D	pointer to SFT we're working with
TCBValSec	+104	4	D	Number of valid (previously written) sectors
TCBpRTCB	+108	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+10c	2	W	process ID for file sharing checks
TCBUser_ID	+10e	2	W	user ID for file sharing checks
TCBfSharing	+110	1	B	non-zero ==> no redirection
TCBSrvAttrib	+111	1	B	see SetAttrib/file.asm
TCBJfnFlag	+112	1	B	JFN flag bits for current fil handle
TCBAllowed	+113	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+114	4	D	server's per file cookie
TCBOpFlags	+118	2	W	whether server wants oplock, etc.
TCBCurBuf	+11a	4	D	currently assigned buffer
TCBThishVPB	+11e	2	W	handle of current VPB
TCBNextAdd	+120	2	W	
TCBBytSecPos	+122	2	W	position of first byte within sector
TCBClusNum	+124	2	W	
TCBLastPos	+126	2	W	
TCBBytCnt1	+128	2	W	Number of bytes in 1st sector
TCBBytCnt2	+12a	2	W	# of bytes in last sector
TCBSecCnt	+12c	2	W	number of whole sectors
TCBSecClusPos	+12e	1	B	posit of first sector within cluster
TCBNoSetDir	+12f	1	B	If TRUE, do not set directory
TCBJoins	+130	1	B	number of joins

TCBPad	+131	1	B	
TCBDevFCB	+132	1	B	Uses Name1, Name2, combined
TCB_direntry	+133	20	S	Directory entry
dir_name	+133	b	B	File name
dir_attr	+13e	1	B	Attribute bits
dir_pad	+13f	8	B	reserved
dir_EAhandle	+147	2	W	First cluster of extended attribute
dir_time	+149	2	W	Time of last write
dir_date	+14b	2	W	Date of last write
dir_firstfile	+14d	2	W	First allocation unit of file
dir_size_l	+14f	2	W	Low 16 bits of file size
dir_size_h	+151	2	W	High 16 bits of file size
TCBName1	+153	c	B	File name buffer *REDIR*
TCBName2	+15f	d	B	
TCBDESTSTART	+16c	2	W	
TCBDirPad	+16e	5	B	
TCBBufHE	+173	1	B	How to handle a HardError
TCBactBufHE	+174	1	B	action response from user on HardErr
TCBfIOLock	+175	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+176	C	S	Lock handle of user mem
TCBThisCDS	+182	4	D	Address of current CDS
TCBThisFSC	+186	4	D	address of current FSC
TCBpTmpCDS	+18a	4	D	Address of dummycds
TCBpOpenBuf	+18e	2	W	Address of current OpenBuf
TCBpSearchBuf	+190	2	W	Address of SearchBuf
TCBFailErr	+192	2	W	NZ if user did FAIL on I 24
TCBShareRetriesLeft	+194	2	W	number of share/lock viol retries
TCBRetryCount	+196	2	W	num of share/lock retries to do
TCBRetryLoop	+198	2	W	num of share/lock retry delay loops
TCB_pSrchBuf	+19a	2	W	internal search buffer
TCB_pOpenBuf	+19c	2	W	Pointer to a scratch buffer on stack
TCBAttrib	+19e	2	W	storage for file attributes *REDIR*
TCBExtFCB	+1a0	1	B	Extended FCB
TCBPad2	+1a1	1	B	
TCBWFP_Start	+1a2	2	W	TASKAREA offset for working string *REDIR*
TCBRen_WFP	+1a4	2	W	WFB pointer for rename destination *REDIR*
TCBWFP_Path_End	+1a6	2	W	End of Path component of string
TCBCurr_Dir_End	+1a8	2	W	

TCBDTAddr	+1aa	4	D	User's I/O transfer address *REDIR*
TCBVolID	+1ae	1	B	!0 if vol ID found in dir search
TCBSpaceFlag	+1af	1	B	Embedded spaces allowed in FCB
TCBCreating	+1b0	1	B	
TCBDelAll	+1b1	1	B	
TCBFoundDel	+1b2	1	B	
TCBFound_dev	+1b3	1	B	true => search found a device 3.10
TCBfSplice	+1b4	1	B	true => do a splice in transpath 3.10
TCBclusFac	+1b5	1	B	sectors/cluster used in dir search
TCBcMeta	+1b6	1	B	components found 3.10
TCBPathNameType	+1b7	1	B	
TCBDevPt	+1b8	4	D	Address of device found by DevName *REDIR*
TCBDirSec	+1bc	4	D	Variables used in directory searching
TCBDirStart	+1c0	2	W	Variables used in directory searching
TCBNxtClusNum	+1c2	2	W	Variables used in directory searching
TCBEntFree	+1c4	2	W	Variables used in directory searching
TCBEntLast	+1c6	2	W	Variables used in directory searching
TCBLastEnt	+1c8	2	W	Variables used in directory searching
TCBSattrib	+1ca	2	W	Storage for search attrs *REDIR* 3.10
TCB_SemInfo	+1cc	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+1d0	4	D	debugger display address for ksems
TCB_NPX_Buffer	+1d4	4	D	
TCBpTCBWaitNext	+1d8	4	D	Next waiting TCB
TCBpTCBWaitList	+1dc	4	D	Threads waiting for me to die
TCBQState	+1e0	1	B	Scheduler queue location (actual)
TCBState	+1e1	1	B	Current scheduler state (desired)
TCBWakeFlags	+1e2	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+1e3	1	B	Window Boost count
TCBPriClass	+1e4	1	B	Priority Class (user)
TCBPriLevel	+1e5	1	B	Priority Level (user)
TCBPriClassMod	+1e6	1	B	Priority Class modifier bits
TCBSchFlags	+1e7	1	B	Misc. Scheduler flags
TCBPriority	+1e8	2	W	Calculated Priority
TCBPriorityMin	+1ea	2	W	Minimum Scheduling priority
TCBcBoostLock	+1ec	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+1f0	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+1f4	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+1f8	4	D	Higher priority thread
TCBpTCBPriLower	+1fc	4	D	Lower priority thread

TCBpTCBPriNext	+200	4	D	Next same-priority thread
TCBpTCBPriPrev	+204	4	D	Prev same-priority thread
TCBpTCBWakeup	+208	4	D	TKQueryWakeup TCB list
TCBSleepID	+20c	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+210	14	S	Timeout/Starvation Timeout element
TCBCheckedSig	+224	1	B	Used by the loader
TCBfSwapping	+225	1	B	status of swapping
TCBVolIONest	+226	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+227	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+228	4	D	I/O request packet for thread
TCBSysTime	+22c	4	D	time spent in system code
TCBUserTime	+230	4	D	time spent in user code
TCB_pPVDBThd	+234	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flgDbg	+238	4	D	
TCBCpl2_ESP	+23c	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+240	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+242	1	B	Value copied from ptda_NewFiles
TCBEntryActions	+243	1	B	Kernel entry force flags
TCBSig_pend	+244	2	W	bit vector of pending signals
TCBSig_holding	+246	2	W	bit vector of postponed signals
TCBSig_cur	+248	2	W	bit vec of signals being processed
TCBXcptRepRec	+24a	4	D	report record of active exception
TCBSig_termtid	+24e	2	W	tid of terminator -75797
TCBSecbits	+250	1	B	Security bits 54735
TCBspbytes	+251	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+252	4	D	Last semaphore cleared in MUX 72485
TCBMiscFlags	+256	1	B	Used for hard error processing
TCBModeFlags	+257	2	W	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+259	1	B	Spare flags
TCBLibiFlags	+25a	1	B	84537
TCBFiller	+25b	1	B	To keep size 4*N
TCB_ProcNameBuf	+25c	4	D	Pointer to procedure name
TCB_ObjNameBuf	+260	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+264	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+268	4	D	Used by loader
TCB_FaultBuf	+26c	4	D	
TCB_ObjNameBufL	+270	2	W	Length of object name buffer
TCB_TmpNameBufL	+272	2	W	
TCB_SrcModNameBufL	+274	2	W	

TCB_FaultBufL	+276	2	W	
TCBSecchild	+278	4	D	Child Security data 54735

#### TCBForcedActions flag definitions:

Name	Bit Mask	Description
TK_FF_BUF	0x00000001	Buffer must be released
TK_FF_EXIT	0x00000002	Call TKExit (old FF_DES)
TK_FF_CRITSEC	0x00000004	Enter Per-task critical section
TK_FF_ICE	0x00000008	Freeze thread
TK_FF_NPX	0x00000010	NPX Error
TK_FF_TIB	0x00000020	Update the TIB
TK_FF_TRC	0x00000040	Enter Debug
TK_FF_SIG	0x00000080	Signal pending
TK_FF_CTXH	0x00000100	Pending local context hooks
TK_FF_STIH	0x00000200	Execute STI hooks
TK_FF_VDMBP	0x00000400	Execute VDM BP hooks
TK_FF_RTRY	0x00000800	Retry V86 system call
TK_FF_PIB	0x00001000	Update the PIB
TK_FF_SCH	0x00002000	Do Scheduler Processing
TK_FF_TFBIT	0x00004000	Validate user eflags TF bit
TK_FF_TIBPRI	0x00008000	Update only the priority fields in TIB 59463

#### TCBEntryActions flag definitions:

Name	Bit Mask	Description
TK_EF_PFCLI	1	Page fault inside CLI
TK_EF_TRC	2	DosDebug action pending

#### TCBWakeFlags flag definitions:

Name	Bit Mask	Description
TK_WF_INTERRUPTED	0x01	Sleep was interrupted
TK_WF_TIMEEXP	0x02	Timeout expired
TK_WF_INTPENDING	0x04	Interrupt pending
TK_WF_SINGLEWAKEUP	0x08	Thread wants single wakeup
TK_WF_INTERRUPTIBLE	0x10	Thread blocked interruptibly
TK_WF_TIMEOUT	0x20	Thread blocked with timeout
TK_WF_SLEEPING	0x40	In TKSleep()

**TCBState and TCBQState definitions:**

Name	Value	Description
STATE_VOID	0	Uninitialized
STATE_READY	1	Ready to run
STATE_BLOCKED	2	Blocked on an ID
STATE_SUSPENDED	3	Suspended (DosSuspendThread)
STATE_CRITSEC	4	Blocked by another CritSec thread
STATE_RUNNING	5	Thread currently running
STATE_READYBOOST	6	Ready, but apply an IO boost
STATE_TSD	7	Thread waiting for TSD
STATE_DELAYED	8	Delayed TKWakeup (Almost Ready)
STATE_FROZEN	9	Frozen Thread (FF_ICE)
STATE_GETSTACK	10	Incomming TSD
STATE_BADSTACK	11	TSD failed to swap in

**TCBPriClassMod definitions:**

Name	Value	Description
CLASSMOD_KEYBOARD	0x04	Keyboard boost
CLASSMOD_STARVED	0x08	Starvation boost
CLASSMOD_DEVICE	0x10	Device I/O Done Boost
CLASSMOD_FOREGROUND	0x20	Foreground boost
CLASSMOD_WINDOW	0x40	Window Boost
CLASSMOD_VDM_INTERRUPT	0x80	VDM simulated interrupt boost

**TCBPriClass definitions:**

Name	Value	Description
CLASS_NOCHANGE	0x00	No priority class change
CLASS_IDLE_TIME	0x01	Idle-Time class
CLASS_REGULAR	0x02	Regular class
CLASS_TIME_CRITICAL	0x03	Time-Critical class
CLASS_SERVER	0x04	Client-Server Server class

**TCBSchFlg flag definitions:**

Name	Bit Mask	Description
SCH_PROTECTED_PRI	0x0001	Only Intra-process SetPri allowed
SCH_WINDOWBOOST_LOCK	0x0002	Lock out windoboost changes

SCH_MINSLICE	0x0004	Use minimum timeslice
SCH_PAGE_FAULT	0x0008	Dynamic timeslicing ###
SCH_PAGE_FAULT_BIT	0x03	Dynamic timeslicing P728371

#### TCBfSwapping flag definitions:

Name	Bit Mask	Description
SM_TCB_SWAPPING	0x01	swap i/o underway
SM_TCB_RESIZING	0x02	data structures are growing

#### TCBMiscFlags flag definitions:

Name	Bit Mask	Description
TMF_CMapFailed	(0x01)	Set if alloc/realloc failed on a cluster map (mft_selCMap).
TMF_IGNORE_HE	(0x02)	If set, ignore (auto fail) hard error
TMF_MULT_XCPT	(0x04)	Set if multiple ring 0 exceptions
TMF_NoFwd	(0x08)	Set if inhibiting forwarders
TMF_EXIT_TERM	(0x10)	TK_FF_EXIT means TKTermThread
TMF_NO_EXCEPT	(0x20)	Indicates TIB exception field invalid
TMF_XCPT_HE	(0x40)	Indicates an exception harderr is pending

#### TCBMSpareFlags flag definitions:

Name	Bit Mask	Description
SPFLAGS_FGND_DISKIO	0x0080	Foreground Disk I/O

#### TCBReqPktFlg flag definitions:

Name	Bit Mask	Description
TK_RP_ALLOCATED	0x01	
TK_RP_INUSE	0x02	

-----

## Thread Control Block for OS/2 Warp V3.0

Field Name	Offset	Length	Type	Description
------------	--------	--------	------	-------------



TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBTLMA	+74	80	D	Thread local memory area
TCBDMAAdd	+f4	4	D	User's I/O transfer address
TCBSecPos	+f8	4	D	Position of first sector accessed within file
TCBThisSFT	+fc	4	D	pointer to SFT we're working with
TCBValSec	+100	4	D	Number of valid (previously written) sectors
TCBpRTCB	+104	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+108	2	W	process ID for file sharing checks
TCBUser_ID	+10a	2	W	user ID for file sharing checks
TCBfSharing	+10c	1	B	non-zero ==> no redirection
TCBSrvAttrib	+10d	1	B	see SetAttrib/file.asm

TCBJfnFlag	+10e	1	B	JFN flag bits for current fil handle
TCBAllowed	+10f	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+110	4	D	server's per file cookie
TCBOpFlags	+114	2	W	whether server wants oplock, etc.
TCBCurBuf	+116	4	D	currently assigned buffer
TCBThishVPB	+11a	2	W	handle of current VPB
TCBNextAdd	+11c	2	W	
TCBBytSecPos	+11e	2	W	position of first byte within sector
TCBClusNum	+120	2	W	
TCBLastPos	+122	2	W	
TCBBytCnt1	+124	2	W	Number of bytes in 1st sector
TCBBytCnt2	+126	2	W	# of bytes in last sector
TCBSecCnt	+128	2	W	number of whole sectors
TCBSecClusPos	+12a	1	B	posit of first sector within cluster
TCBBufHE	+12b	1	B	How to handle a HardError
TCBactBufHE	+12c	1	B	action response from user on HardErr
TCBfIOLock	+12d	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+12e	C	S	Lock handle of user mem
TCBThisCDS	+13a	4	D	Address of current CDS
TCBThisFSC	+13e	4	D	address of current FSC
TCBpTmpCDS	+142	4	D	Address of dummycds
TCBpOpenBuf	+146	2	W	Address of current OpenBuf
TCBpSearchBuf	+148	2	W	Address of SearchBuf
TCBFailErr	+14a	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+14c	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+150	4	D	debugger display address for ksems
TCB_NPX_Buffer	+154	4	D	
TCBpTCBWaitNext	+158	4	D	Next waiting TCB
TCBpTCBWaitList	+15c	4	D	Threads waiting for me to die
TCBQState	+160	1	B	Scheduler queue location (actual)
TCBState	+161	1	B	Current scheduler state (desired)
TCBWakeFlags	+162	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+163	1	B	Window Boost count
TCBPriClass	+164	1	B	Priority Class (user)
TCBPriLevel	+165	1	B	Priority Level (user)
TCBPriClassMod	+166	1	B	Priority Class modifier bits
TCBSchFlags	+167	1	B	Misc. Scheduler flags
TCBPriority	+168	2	W	Calculated Priority
TCBPriorityMin	+16a	2	W	Minimum Scheduling priority

TCBcBoostLock	+16c	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+170	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+174	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+178	4	D	Higher priority thread
TCBpTCBPriLower	+17c	4	D	Lower priority thread
TCBpTCBPriNext	+180	4	D	Next same-priority thread
TCBpTCBPriPrev	+184	4	D	Prev same-priority thread
TCBpTCBWakeup	+188	4	D	TKQueryWakeup TCB list
TCBSleepID	+18c	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+190	10	S	Timeout/Starvation Timeout element
TCBCheckedSig	+1a0	1	B	Used by the loader
TCBfSwapping	+1a1	1	B	status of swapping
TCBVolIONest	+1a2	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+1a3	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+1a4	4	D	I/O request packet for thread
TCBSysTime	+1a8	4	D	time spent in system code
TCBUserTime	+1ac	4	D	time spent in user code
TCB_pPVDBThd	+1b0	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+1b4	4	D	
TCBCpl2_ESP	+1b8	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+1bc	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+1be	1	B	Value copied from ptda_NewFiles
TCBEntryActions	+1bf	1	B	Kernel entry force flags
TCBSig_pend	+1c0	2	W	bit vector of pending signals
TCBSig_holding	+1c2	2	W	bit vector of postponed signals
TCBSig_cur	+1c4	2	W	bit vec of signals being processed
TCBXcptRepRec	+1c6	4	D	report record of active exception
TCBSig_termtid	+1ca	2	W	tid of terminator -75797
TCBSecbits	+1cc	1	B	Security bits 54735
TCBspbytes	+1cd	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+1ce	4	D	Last semaphore cleared in MUX 72485
TCBMiscFlags	+1d2	1	B	Used for hard error processing
TCBModeFlags	+1d3	2	W	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+1d5	1	B	Spare flags
TCBLibiFlags	+1d6	1	B	84537
TCBFiller	+1d7	1	B	To keep size 4*N
TCB_ProcNameBuf	+1d8	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1dc	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1e0	4	D	aka TCB_TgtModNameBuf

TCB_SrcModNameBuf	+1e4	4	D	Used by loader
TCB_FaultBuf	+1e8	4	D	
TCB_ObjNameBufL	+1ec	2	W	Length of object name buffer
TCB_TmpNameBufL	+1ee	2	W	
TCB_SrcModNameBufL	+1f0	2	W	
TCB_FaultBufL	+1f2	2	W	
TCBSecchild	+1f4	4	D	Child Security data 54735

-----

## Thread Control Block for OS/2 Warp V3.0 with Fix-Pack

See: [Fix pack 09](#) for details of the change introduced in this fix-pack.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA

TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBpDHRetAddr	+74	4	D	82818 Pointer to DHRouter return address
TCBTLMA	+78	80	D	Thread local memory area
TCBDMAAdd	+f8	4	D	User's I/O transfer address
TCBSecPos	+fc	4	D	Position of first sector accessed within file
TCBThisSFT	+100	4	D	pointer to SFT we're working with
TCBValSec	+104	4	D	Number of valid (previously written) sectors
TCBpRTCB	+108	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+10c	2	W	process ID for file sharing checks
TCBUser_ID	+10e	2	W	user ID for file sharing checks
TCBfSharing	+110	1	B	non-zero ==> no redirection
TCBSrvAttrib	+111	1	B	see SetAttrib/file.asm
TCBJfnFlag	+112	1	B	JFN flag bits for current fil handle
TCBAllowed	+113	1	B	Allowed I 24 answers (see allowed_)
TCBpCookie	+114	4	D	server's per file cookie
TCBpFlags	+118	2	W	whether server wants oplock, etc.
TCBCurBuf	+11a	4	D	currently assigned buffer
TCBThishVPB	+11e	2	W	handle of current VPB
TCBNextAdd	+120	2	W	
TCBBytSecPos	+122	2	W	position of first byte within sector
TCBclusNum	+124	2	W	
TCBLastPos	+126	2	W	
TCBBytCnt1	+128	2	W	Number of bytes in 1st sector
TCBBytCnt2	+12a	2	W	# of bytes in last sector
TCBSecCnt	+12c	2	W	number of whole sectors
TCBSecClusPos	+12e	1	B	posit of first sector within cluster
TCBBufHE	+12f	1	B	How to handle a HardError
TCBactBufHE	+130	1	B	action response from user on HardErr
TCBfIOLock	+131	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+132	C	S	Lock handle of user mem
TCBThisCDS	+13e	4	D	Address of current CDS
TCBThisFSC	+142	4	D	address of current FSC
TCBpTmpCDS	+146	4	D	Address of dummycds
TCBpOpenBuf	+14a	2	W	Address of current OpenBuf
TCBpSearchBuf	+14c	2	W	Address of SearchBuf

TCBFailErr	+14e	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+150	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+154	4	D	debugger display address for ksems
TCB_NPX_Buffer	+158	4	D	
TCBpTCBWaitNext	+15c	4	D	Next waiting TCB
TCBpTCBWaitList	+160	4	D	Threads waiting for me to die
TCBQState	+164	1	B	Scheduler queue location (actual)
TCBState	+165	1	B	Current scheduler state (desired)
TCBWakeFlags	+166	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+167	1	B	Window Boost count
TCBPriClass	+168	1	B	Priority Class (user)
TCBPriLevel	+169	1	B	Priority Level (user)
TCBPriClassMod	+16a	1	B	Priority Class modifier bits
TCBSchFlags	+16b	1	B	Misc. Scheduler flags
TCBPriority	+16c	2	W	Calculated Priority
TCBPriorityMin	+16e	2	W	Minimum Scheduling priority
TCBcBoostLock	+170	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+174	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+178	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+17c	4	D	Higher priority thread
TCBpTCBPriLower	+180	4	D	Lower priority thread
TCBpTCBPriNext	+184	4	D	Next same-priority thread
TCBpTCBPriPrev	+188	4	D	Prev same-priority thread
TCBpTCBWakeup	+18c	4	D	TKQueryWakeup TCB list
TCBSleepID	+190	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+194	10	S	Timeout/Starvation Timeout element
TCBCheckedSig	+1a4	1	B	Used by the loader
TCBfSwapping	+1a5	1	B	status of swapping
TCBVolIONest	+1a6	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+1a7	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+1a8	4	D	I/O request packet for thread
TCBSysTime	+1ac	4	D	time spent in system code
TCBUserTime	+1b0	4	D	time spent in user code
TCB_pPVDBThd	+1b4	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+1b8	4	D	
TCBCpl2_ESP	+1bc	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+1c0	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+1c2	1	B	Value copied from ptda_NewFiles
TCBEntryActions	+1c3	1	B	Kernel entry force flags

TCBSig_pend	+1c4	2	W	bit vector of pending signals
TCBSig_holding	+1c6	2	W	bit vector of postponed signals
TCBSig_cur	+1c8	2	W	bit vec of signals being processed
TCBXcptRepRec	+1ca	4	D	report record of active exception
TCBSig_termtid	+1ce	2	W	tid of terminator -75797
TCBSecbits	+1d0	1	B	Security bits 54735
TCBspbytes	+1d1	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+1d2	4	D	Last semaphore cleared in MUX 72485
TCBMiscFlags	+1d6	1	B	Used for hard error processing
TCBModeFlags	+1d7	2	W	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+1d9	1	B	Spare flags
TCBLibiFlags	+1da	1	B	84537
TCBFiller	+1db	1	B	To keep size 4*N
TCB_ProcNameBuf	+1dc	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1e0	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1e4	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1e8	4	D	Used by loader
TCB_FaultBuf	+1ec	4	D	
TCB_ObjNameBufL	+1f0	2	W	Length of object name buffer
TCB_TmpNameBufL	+1f2	2	W	
TCB_SrcModNameBufL	+1f4	2	W	
TCB_FaultBufL	+1f6	2	W	
TCBSecchild	+1f8	4	D	Child Security data 54735

#### TCBLibiFlags flag definitions:

Name	Bit Mask	Description
INIT_ROUTINE_FAILED	(0x01)	84537 Set if dll init routine failed

-----

## Thread Control Block for OS/2 Warp V3.0 with Fix-Pack 11 or Later

Field Name	Offset	Length	Type	Description
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number

TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBpDHRetAddr	+74	4	D	82818 Pointer to DHRouter return address
TCBTLMA	+78	80	D	Thread local memory area
TCBDMAAdd	+f8	4	D	User's I/O transfer address
TCBSecPos	+fc	4	D	Position of first sector accessed within file
TCBThisSFT	+100	4	D	pointer to SFT we're working with
TCBValSec	+104	4	D	Number of valid (previously written) sectors
TCBpRTCB	+108	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+10c	2	W	process ID for file sharing checks
TCBUser_ID	+10e	2	W	user ID for file sharing checks
TCBfSharing	+110	1	B	non-zero ==> no redirection
TCBSrvAttrib	+111	1	B	see SetAttrib/file.asm
TCBJfnFlag	+112	1	B	JFN flag bits for current fil handle



TCBAllowed	+113	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+114	4	D	server's per file cookie
TCBOpFlags	+118	2	W	whether server wants oplock, etc.
TCBCurBuf	+11a	4	D	currently assigned buffer
TCBThishVPB	+11e	2	W	handle of current VPB
TCBNextAdd	+120	2	W	
TCBBytSecPos	+122	2	W	position of first byte within sector
TCBClusNum	+124	2	W	
TCBLastPos	+126	2	W	
TCBBytCnt1	+128	2	W	Number of bytes in 1st sector
TCBBytCnt2	+12a	2	W	# of bytes in last sector
TCBSecCnt	+12c	2	W	number of whole sectors
TCBSecClusPos	+12e	1	B	posit of first sector within cluster
TCBBufHE	+12f	1	B	How to handle a HardError
TCBactBufHE	+130	1	B	action response from user on HardErr
TCBfIOLock	+131	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+132	C	S	Lock handle of user mem
TCBThisCDS	+13e	4	D	Address of current CDS
TCBThisFSC	+142	4	D	address of current FSC
TCBpTmpCDS	+146	4	D	Address of dummycds
TCBpOpenBuf	+14a	2	W	Address of current OpenBuf
TCBpSearchBuf	+14c	2	W	Address of SearchBuf
TCBFailErr	+14e	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+150	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+154	4	D	debugger display address for ksems
TCB_NPX_Buffer	+158	4	D	
TCBpTCBWaitNext	+15c	4	D	Next waiting TCB
TCBpTCBWaitList	+160	4	D	Threads waiting for me to die
TCBQState	+164	1	B	Scheduler queue location (actual)
TCBState	+165	1	B	Current scheduler state (desired)
TCBWakeFlags	+166	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+167	1	B	Window Boost count
TCBPriClass	+168	1	B	Priority Class (user)
TCBPriLevel	+169	1	B	Priority Level (user)
TCBPriClassMod	+16a	1	B	Priority Class modifier bits
TCBSchFlags	+16b	1	B	Misc. Scheduler flags
TCBPriority	+16c	2	W	Calculated Priority
TCBPriorityMin	+16e	2	W	Minimum Scheduling priority
TCBcBoostLock	+170	4	D	Kernel Boost Lock nesting count.

TCBpTCBPriNextQ	+174	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+178	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+17c	4	D	Higher priority thread
TCBpTCBPriLower	+180	4	D	Lower priority thread
TCBpTCBPriNext	+184	4	D	Next same-priority thread
TCBpTCBPriPrev	+188	4	D	Prev same-priority thread
TCBpTCBWakeUp	+18c	4	D	TKQueryWakeUp TCB list
TCBSleepID	+190	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+194	14	S	Timeout/Starvation Timeout element
TCBCheckedSig	+1a8	1	B	Used by the loader
TCBfSwapping	+1a9	1	B	status of swapping
TCBVolIONest	+1aa	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+1ab	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+1ac	4	D	I/O request packet for thread
TCBSysTime	+1b0	4	D	time spent in system code
TCBUserTime	+1b4	4	D	time spent in user code
TCB_pPVDBThd	+1b8	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_fldbgb	+1bc	4	D	
TCBCpl2_ESP	+1c0	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+1c4	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+1c6	1	B	Value copied from ptda_NewFiles
TCBEntryActions	+1c7	1	B	Kernel entry force flags
TCBSig_pend	+1c8	2	W	bit vector of pending signals
TCBSig_holding	+1ca	2	W	bit vector of postponed signals
TCBSig_cur	+1cc	2	W	bit vec of signals being processed
TCBXcptRepRec	+1ce	4	D	report record of active exception
TCBSig_terminator	+1d2	2	W	tid of terminator -75797
TCBSecbits	+1d4	1	B	Security bits 54735
TCBspbytes	+1d5	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+1d6	4	D	Last semaphore cleared in MUX 72485
TCBMiscFlags	+1da	1	B	Used for hard error processing
TCBModeFlags	+1db	2	W	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+1dd	1	B	Spare flags
TCBLibiFlags	+1de	1	B	84537
TCBFiller	+1df	1	B	To keep size 4*N
TCB_ProcNameBuf	+1e0	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1e4	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1e8	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1ec	4	D	Used by loader

TCB_FaultBuf	+1f0	4	D	
TCB_ObjNameBufL	+1f4	2	W	Length of object name buffer
TCB_TmpNameBufL	+1f6	2	W	
TCB_SrcModNameBufL	+1f8	2	W	
TCB_FaultBufL	+1fa	2	W	
TCBSecchild	+1fc	4	D	Child Security data 54735

-----

## Thread Control Block for OS/2 V2.11 with Fix-Pack 90 or Later

See: [Fix pack 90](#) for details of the change introduced in this fix-pack.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number
TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA

TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBpDHRetAddr	+74	4	D	82818 Pointer to DHRouter return address
TCBDMAAdd	+78	4	D	User's I/O transfer address
TCBSecPos	+7c	4	D	Position of first sector accessed within file
TCBThisSFT	+80	4	D	pointer to SFT we're working with
TCBValSec	+84	4	D	Number of valid (previously written) sectors
TCBpRTCB	+88	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+8c	2	W	process ID for file sharing checks
TCBUser_ID	+8e	2	W	user ID for file sharing checks
TCBfSharing	+90	1	B	non-zero ==> no redirection
TCBSrvAttrib	+91	1	B	see SetAttrib/file.asm
TCBJfnFlag	+92	1	B	JFN flag bits for current fil handle
TCBAllowed	+93	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+94	4	D	server's per file cookie
TCBOpFlags	+98	2	W	whether server wants oplock, etc.
TCBCurBuf	+9a	4	D	currently assigned buffer
TCBThishVPB	+9e	2	W	handle of current VPB
TCBNextAdd	+a0	2	W	
TCBBytSecPos	+a2	2	W	position of first byte within sector
TCBClusNum	+a4	2	W	
TCBLastPos	+a6	2	W	
TCBBytCnt1	+a8	2	W	Number of bytes in 1st sector
TCBBytCnt2	+aa	2	W	# of bytes in last sector
TCBSecCnt	+ac	2	W	number of whole sectors
TCBSecClusPos	+ae	1	B	posit of first sector within cluster
TCBBufHE	+af	1	B	How to handle a HardError
TCBactBufHE	+b0	1	B	action response from user on HardErr
TCBfIOLock	+b1	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+b2	C	S	Lock handle of user mem
TCBThisCDS	+be	4	D	Address of current CDS
TCBThisFSC	+c2	4	D	address of current FSC
TCBpTmpCDS	+c6	4	D	Address of dummycds
TCBpOpenBuf	+ca	2	W	Address of current OpenBuf
TCBpSearchBuf	+cc	2	W	Address of SearchBuf
TCBFailErr	+ce	2	W	NZ if user did FAIL on I 24

TCB_SemInfo	+d0	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+d4	4	D	debugger display address for ksems
TCB_NPX_Buffer	+d8	4	D	
TCBpTCBWaitNext	+dc	4	D	Next waiting TCB
TCBpTCBWaitList	+e0	4	D	Threads waiting for me to die
TCBQState	+e4	1	B	Scheduler queue location (actual)
TCBState	+e5	1	B	Current scheduler state (desired)
TCBWakeFlags	+e6	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+e7	1	B	Window Boost count
TCBPriClass	+e8	1	B	Priority Class (user)
TCBPriLevel	+e9	1	B	Priority Level (user)
TCBPriClassMod	+ea	1	B	Priority Class modifier bits
TCBSchFlags	+eb	1	B	Misc. Scheduler flags
TCBPriority	+ec	2	W	Calculated Priority
TCBPriorityMin	+ee	2	W	Minimum Scheduling priority
TCBcBoostLock	+f0	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+f4	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+f8	4	D	Previous priority queue in chain
TCBpTCBPriHigher	+fc	4	D	Higher priority thread
TCBpTCBPriLower	+100	4	D	Lower priority thread
TCBpTCBPriNext	+104	4	D	Next same-priority thread
TCBpTCBPriPrev	+108	4	D	Prev same-priority thread
TCBpTCBWakeup	+10c	4	D	TKQueryWakeup TCB list
TCBSleepID	+110	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+114	14	S	Timeout/Starvation Timeout element
TCBCheckedSig	+128	1	B	Used by the loader
TCBfSwapping	+129	1	B	status of swapping
TCBVolIONest	+12a	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+12b	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+12c	4	D	I/O request packet for thread
TCBpMemStatCur	+130	4	D	Current structure being filled in
TCBMemStat	+134	3C	S	statistics structure
TCBSysTime	+170	4	D	time spent in system code
TCBUserTime	+174	4	D	time spent in user code
TCB_pPVDBThd	+178	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_fldbgb	+17c	4	D	
TCBCpl2_ESP	+180	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+184	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+186	1	W	Value copied from ptda_NewFiles

TCBEntryActions	+187	1	B	Kernel entry force flags
TCBSig_pend	+188	2	W	bit vector of pending signals
TCBSig_holding	+18a	2	W	bit vector of postponed signals
TCBSig_cur	+18c	2	W	bit vec of signals being processed
TCBXcptRepRec	+18e	4	D	report record of active exception
TCBSig_termtid	+192	2	W	
TCBSecbits	+194	1	B	Security bits 54735
TCBspbytes	+195	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+196	4	D	
TCBMiscFlags	+19a	1	D	Used for hard error processing
TCBModeFlags	+19b	2	D	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+19d	1	B	Spare flags
TCBLibiFlags	+19e	1	B	
TCBFiller	+19f	1	B	
TCB_ProcNameBuf	+1a0	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1a4	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1a8	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1ac	4	D	Used by loader
TCB_FaultBuf	+1b0	4	D	
TCB_ObjNameBufL	+1b4	2	W	Length of object name buffer
TCB_TmpNameBufL	+1b6	2	W	
TCB_SrcModNameBufL	+1b8	2	W	
TCB_FaultBufL	+1ba	2	W	
TCBSecchild	+1bc	4	D	Child Security data 54735

#### TCBLibiFlags flag definitions:

Name	Bit Mask	Description
INIT_ROUTINE_FAILED	(0x01)	84537 Set if dll init routine failed

-----

## Thread Control Block for OS/2 V2.11

Field Name	Offset	Length	Type	Description
TCBOrdinal	+0	2	W	Ordinal number of thread in PTDA
TCBNumber	+2	2	W	Thread slot number

TCBForcedActions	+4	4	D	Bit vector of forced actions
TCBpPTDA	+8	4	D	Pointer to the PTDA
TCBpTSD	+c	4	D	Pointer to thread swappable data
TCBptib	+10	4	D	Pointer to thread info block
TCBpTCBNext	+14	4	D	forward link to next (active) TCB
TCBcbStackMax	+18	4	D	Virtual size of stack object
TCBcbStackCur	+1c	4	D	Committed size of stack object
TCBpStack	+20	4	D	Virtual base of stack
TCBpStack16Lo	+24	4	D	Virtual base of 16-bit stack
TCBpStack16Hi	+28	4	D	Virtual limit of 16-bit stack
TCBpLibiHead	+2c	4	D	Link to libi load data area
TCBpLibiCurr	+30	4	D	Link to libi load data area
TCBpLibiFree	+34	4	D	Link to libi free data area
TCB_pcriFrameType	+38	4	D	stack frame type
TCB_pFrameBase	+3c	4	D	stack frame base pointer
TCB_hookheadLocal	+40	8	D	local context hook head
TCB_phookOwnerHead	+48	4	D	linked list of hook blocks
TCBpteKStackTCB0	+4c	4	D	KStack page 0 of TCB
TCBpteKStackTCB1	+50	4	D	KStack page 1 of TCB
TCBpteKStackTSD	+54	4	D	KStack TSD page
TCBpteKStackPTDA0	+58	4	D	KStack page 0 of PTDA
TCBpteKStackPTDA1	+5c	4	D	KStack page 1 of PTDA
TCBpteKStackPTDA2	+60	4	D	KStack page 2 of PTDA
TCBCurrTCB	+64	4	D	SS-relative offset of Current TCB
TCBCurrTSD	+68	4	D	SS-relative offset of Current TSD
TCBBiasTCB	+6c	4	D	stack-to-flat TCB conversion value
TCBBiasTSD	+70	4	D	stack-to-flat TSD conversion value
TCBDMAAdd	+74	4	D	User's I/O transfer address
TCBSecPos	+78	4	D	Position of first sector accessed within file
TCBThisSFT	+7c	4	D	pointer to SFT we're working with
TCBValSec	+80	4	D	Number of valid (previously written) sectors
TCBpRTCB	+84	4	D	Redirector TCB (Used by LANMAN)
TCBProc_ID	+88	2	W	process ID for file sharing checks
TCBUser_ID	+8a	2	W	user ID for file sharing checks
TCBfSharing	+8c	1	B	non-zero ==> no redirection
TCBSrvAttrib	+8d	1	B	see SetAttrib/file.asm
TCBJfnFlag	+8e	1	B	JFN flag bits for current fil handle
TCBAllowed	+8f	1	B	Allowed I 24 answers (see allowed_)
TCBOpCookie	+90	4	D	server's per file cookie

TCBOpFlags	+94	2	W	whether server wants oplock, etc.
TCBCurBuf	+96	4	D	currently assigned buffer
TCBThishVPB	+9a	2	W	handle of current VPB
TCBNextAdd	+9c	2	W	
TCBBytSecPos	+9e	2	W	position of first byte within sector
TCBClusNum	+a0	2	W	
TCBLastPos	+a2	2	W	
TCBBytCnt1	+a4	2	W	Number of bytes in 1st sector
TCBBytCnt2	+a6	2	W	# of bytes in last sector
TCBSecCnt	+a8	2	W	number of whole sectors
TCBSecClusPos	+aa	1	B	posit of first sector within cluster
TCBBufHE	+ab	1	B	How to handle a HardError
TCBactBufHE	+ac	1	B	action response from user on HardErr
TCBfIOLock	+ad	1	B	NZ if TCBLockHndl is valid
TCBLockHndl	+ae	C	S	Lock handle of user mem
TCBThisCDS	+ba	4	D	Address of current CDS
TCBThisFSC	+be	4	D	address of current FSC
TCBpTmpCDS	+c2	4	D	Address of dummycds
TCBpOpenBuf	+c6	2	W	Address of current OpenBuf
TCBpSearchBuf	+c8	2	W	Address of SearchBuf
TCBFailErr	+ca	2	W	NZ if user did FAIL on I 24
TCB_SemInfo	+cc	4	D	16bit addr of the ramsem blocked upon
TCB_SemDebugAddr	+d0	4	D	debugger display address for ksems
TCB_NPX_Buffer	+d4	4	D	
TCBpTCBWaitNext	+d8	4	D	Next waiting TCB
TCBpTCBWaitList	+dc	4	D	Threads waiting for me to die
TCBQState	+e0	1	B	Scheduler queue location (actual)
TCBState	+e1	1	B	Current scheduler state (desired)
TCBWakeFlags	+e2	1	B	TKSleep/TKWakeup Flags
TCBcWindowBoost	+e3	1	B	Window Boost count
TCBPriClass	+e4	1	B	Priority Class (user)
TCBPriLevel	+e5	1	B	Priority Level (user)
TCBPriClassMod	+e6	1	B	Priority Class modifier bits
TCBSchFlags	+e7	1	B	Misc. Scheduler flags
TCBPriority	+e8	2	W	Calculated Priority
TCBPriorityMin	+ea	2	W	Minimum Scheduling priority
TCBcBoostLock	+ec	4	D	Kernel Boost Lock nesting count.
TCBpTCBPriNextQ	+f0	4	D	Next priority queue in chain
TCBpTCBPriPrevQ	+f4	4	D	Previous priority queue in chain



TCBpTCBPriHigher	+f8	4	D	Higher priority thread
TCBpTCBPriLower	+fc	4	D	Lower priority thread
TCBpTCBPriNext	+100	4	D	Next same-priority thread
TCBpTCBPriPrev	+104	4	D	Prev same-priority thread
TCBpTCBWakeup	+108	4	D	TKQueryWakeup TCB list
TCBSleepID	+10c	4	D	Sleep ID this TCB is sleeping on
TCBtoe	+110	14	S	Timeout/Starvation Timeout element
TCBCheckedSig	+124	1	B	Used by the loader
TCBfSwapping	+125	1	B	status of swapping
TCBVolIONest	+126	1	B	nesting level of FSH_DoVolIO
TCBReqPktFlg	+127	1	B	Flag to indicate if request pkt in use
TCBReqPkt	+128	4	D	I/O request packet for thread
TCBpMemStatCur	+12c	4	D	Current structure being filled in
TCBMemStat	+130	3C	S	statistics structure
TCBSysTime	+16c	4	D	time spent in system code
TCBUserTime	+170	4	D	time spent in user code
TCB_pPVDBThd	+174	4	D	Ptr to Perfview Data Block for this thread (pvdb_thd_s).
TCB_flDbg	+178	4	D	
TCBCpl2_ESP	+17c	4	D	Saved TSS CPL2 stack pointer.
TCBCpl2_SS	+180	2	W	Saved TSS CPL2 stack segment.
TCBNewFlags	+182	1	W	Value copied from ptda_NewFiles
TCBEntryActions	+183	1	B	Kernel entry force flags
TCBSig_pend	+184	2	W	bit vector of pending signals
TCBSig_holding	+186	2	W	bit vector of postponed signals
TCBSig_cur	+188	2	W	bit vec of signals being processed
TCBXcptRepRec	+18a	4	D	report record of active exception
TCBSig_termtid	+18e	2	W	
TCBSecbits	+190	1	B	Security bits 54735
TCBspbytes	+191	1	B	To keep size 4*N 54735
TCB_ulSRIndex	+192	4	D	
TCBMiscFlags	+196	1	D	Used for hard error processing
TCBModeFlags	+197	2	D	Mode flags for OPEN - for WhatVolume
TCBSpareFlags	+199	1	B	Spare flags
TCBLibiFlags	+19a	1	B	
TCBFiller	+19b	1	B	
TCB_ProcNameBuf	+19c	4	D	Pointer to procedure name
TCB_ObjNameBuf	+1a0	4	D	Pointer to object name buffer
TCB_TmpNameBuf	+1a4	4	D	aka TCB_TgtModNameBuf
TCB_SrcModNameBuf	+1a8	4	D	Used by loader

TCB_FaultBuf	+1ac	4	D	
TCB_ObjNameBufL	+1b0	2	W	Length of object name buffer
TCB_TmpNameBufL	+1b2	2	W	
TCB_SrcModNameBufL	+1b4	2	W	
TCB_FaultBufL	+1b6	2	W	
TCBSecchild	+1b8	4	D	Child Security data 54735

-----

## Thread Swappable Data for OS/2 Warp V4.0 and OS/2 Warp V3.0 ALLSTRICT kernel

For **TSD** formats for other versions of OS/2 see:

**TSD** for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

**TSD** for OS/2 V2.11 ALLSTRICT kernel

**TSD** for OS/2 V2.11 RETAIL kernel

### Pointers

**TCBpTSD** points to the TSD associated with a TCB

**CurrTSD** points to the current TSD.

### Locations

System Arena.

### VM Owner

**tsd (0xffcd)**

### Format

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
TSDpad	+0	1000	B	Dummy page to catch faults
TSDUserStack	+1000	F98	W	Thread's kernel stack
TSDUserESP	+1f98	4	D	Saved user stack pointer
TSDUserSS	+1f9c	2	W	Saved user stack segment
TSDUserSSPad	+1f9e	2	W	Pad word pushed by gate
TSDKernelESP	+1fa0	4	D	Saved kernel stack pointer.
TSDpTCB	+1fa4	4	D	Link to TCB
TSDpfnFault	+1fa8	4	D	ptr to local fault handler in effect
TSDTrapNum	+1fac	4	D	TrapNum from the last fault
TSDerrrcFault	+1fb0	4	D	error code from the last fault
TSDpljmp	+1fb4	4	D	Buffer saved by TKCatchFault
TSDselFault	+1fb8	2	W	faulting selector
TSDCpl2_SSSize	+1fba	2	W	Size of ring 2 stack - atleast thats what the user beleives

TSDdescLDT	+1fbc	8	D	LDT table descriptor
TSDdescKStackSS	+1fc4	8	D	SS descriptor
TSDdescFPEM	+1fcc	8	D	reserved descriptor slot
TSDdescTIB	+1fd4	8	D	FS mapping to TIB
TSDulExitCode	+1fdc	4	D	Proposed Thread Exit code (for dbg)
TSDerridFault	+1fe0	4	D	error id from page fault
TSDPFErr	+1fe4	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+1fe8	4	D	
TSDlDbgRangeEnd	+1fec	4	D	
TSDlDbgLastAddr	+1ff0	4	D	
TSDpPCB	+1ff4	4	D	Pointer to Profile Control Block
TSDpDLLTerm	+1ff8	4	D	Pointer to data buffer
TSDcObjSem	+1ffc	4	D	Count of object semaphores held

-----

## Thread Swappable Data for OS/2 Warp V3.0 RETAIL kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
TSDUserStack	+0	F9C	W	Thread's kernel stack
TSDUserESP	+f9c	4	D	Saved user stack pointer
TSDUserSS	+fa0	2	W	Saved user stack segment
TSDUserSSPad	+fa2	2	W	Pad word pushed by gate
TSDKernelESP	+fa4	4	D	Saved kernel stack pointer.
TSDpTCB	+fa8	4	D	Link to TCB
TSDpfnFault	+fac	4	D	ptr to local fault handler in effect
TSDTrapNum	+fb0	4	D	TrapNum from the last fault
TSDerrcFault	+fb4	4	D	error code from the last fault
TSDpljmp	+fb8	4	D	Buffer saved by TKCatchFault
TSDselFault	+fbc	2	W	faulting selector
TSDCpl2_SSSize	+fbe	2	W	Size of ring 2 stack - atleast thats what the user beleives
TSDdescLDT	+fc0	8	D	LDT table descriptor
TSDdescKStackSS	+fc8	8	D	SS descriptor
TSDdescFPEM	+fd0	8	D	reserved descriptor slot
TSDdescTIB	+fd8	8	D	FS mapping to TIB
TSDulExitCode	+fe0	4	D	Proposed Thread Exit code (for dbg)

TSDerridFault	+fe4	4	D	error id from page fault
TSDPFErr	+fe8	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+fec	4	D	
TSDlDbgRangeEnd	+ff0	4	D	
TSDlDbgLastAddr	+ff4	4	D	
TSDpPCB	+ff8	4	D	Pointer to Profile Control Block
TSDpDLLTerm	+ffc	4	D	Pointer to data buffer

-----

## Thread Swappable Data for OS/2 V2.11 ALLSTRICT kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
TSDpad	+0	1000	B	Dummy page to catch faults
TSDUserStack	+1000	F98	W	Thread's kernel stack
TSDUserESP	+1f98	4	D	Saved user stack pointer
TSDUserSS	+1f9c	2	W	Saved user stack segment
TSDUserSSPad	+1f9e	2	W	Pad word pushed by gate
TSDKernelESP	+1fa0	4	D	Saved kernel stack pointer.
TSDpTCB	+1fa4	4	D	Link to TCB
TSDpfnFault	+1fa8	4	D	ptr to local fault handler in effect
TSDTrapNum	+1fac	4	D	TrapNum from the last fault
TSDerrcFault	+1fb0	4	D	error code from the last fault
TSDpljmp	+1fb4	4	D	Buffer saved by TKCatchFault
TSDselFault	+1fb8	2	W	faulting selector
TSDCpl2_SSSize	+1fba	2	W	Size of ring 2 stack - atleast thats what the user beleives
TSDdescLDT	+1fbc	8	D	LDT table descriptor
TSDdescKStackSS	+1fc4	8	D	SS descriptor
TSDdescFPEM	+1fcc	8	D	reserved descriptor slot
TSDdescTIB	+1fd4	8	D	FS mapping to TIB
TSDulExitCode	+1fdc	4	D	Proposed Thread Exit code (for dbg)
TSDerridFault	+1fe0	4	D	error id from page fault
TSDPFErr	+1fe4	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+1fe8	4	D	
TSDlDbgRangeEnd	+1fec	4	D	
TSDlDbgLastAddr	+1ff0	4	D	
TSDpPCB	+1ff4	4	D	Pointer to Profile Control Block

TSDpDLLTerm	+1ff8	4	D	Pointer to data buffer
TSDcObjSem	+1ffc	4	D	Count of object semaphores held

-----

## Thread Swappable Data for OS/2 V2.11 RETAIL kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
TSDUserStack	+0	F9C	W	Thread's kernel stack
TSDUserESP	+f9c	4	D	Saved user stack pointer
TSDUserSS	+fa0	2	W	Saved user stack segment
TSDUserSSPad	+fa2	2	W	Pad word pushed by gate
TSDKernelESP	+fa4	4	D	Saved kernel stack pointer.
TSDpTCB	+fa8	4	D	Link to TCB
TSDpfnFault	+fac	4	D	ptr to local fault handler in effect
TSDTrapNum	+fb0	4	D	TrapNum from the last fault
TSDerrcFault	+fb4	4	D	error code from the last fault
TSDpljmp	+fb8	4	D	Buffer saved by TKCatchFault
TSDselFault	+fbc	2	W	faulting selector
TSDCpl2_SSSize	+fbe	2	W	Size of ring 2 stack - atleast thats what the user beleives
TSDdescLDT	+fc0	8	D	LDT table descriptor
TSDdescKStackSS	+fc8	8	D	SS descriptor
TSDdescFPEM	+fd0	8	D	reserved descriptor slot
TSDdescTIB	+fd8	8	D	FS mapping to TIB
TSDulExitCode	+fe0	4	D	Proposed Thread Exit code (for dbg)
TSDerridFault	+fe4	4	D	error id from page fault
TSDPFErr	+fe8	4	D	actual error from PGPagefault
TSDlDbgRangeStart	+fec	4	D	
TSDlDbgRangeEnd	+ff0	4	D	
TSDlDbgLastAddr	+ff4	4	D	
TSDpPCB	+ff8	4	D	Pointer to Profile Control Block
TSDpDLLTerm	+ffc	4	D	Pointer to data buffer

-----

## Local Exception Handler Long-Jump Buffer

Pointers

**TSDpljmp** points to current reistered ljmp buffer.

**ljmp\_pljmp** points to the next nested ljmp buffer.

Locations

System Arena. Usually allocated as local data on the Ring 0 stack.

VM Owner

**tsd (0xffcd)**

Format

Field Name	Offset	Length	Type	Description
ljmp_lEBX	+0	4	D	EBX restored when local exception handler returns control
ljmp_lESI	+4	4	D	ESI restored when local exception handler returns control
ljmp_lEDI	+8	4	D	EDI restored when local exception handler returns control
ljmp_lEBP	+c	4	D	EBP restored when local exception handler returns control
ljmp_lESP	+10	4	D	ESP restored when local exception handler returns control
ljmp_lEIP	+14	4	D	EIP restored when local exception handler returns control
ljmp_pfnFault	+18	4	D	Address of previous fault handler
ljmp_pljmp	+1c	4	D	Address of previous long jump buffer

-----

# Per-Task Data Area for OS/2 Warp V4.0 ALLSTRICT kernel

For **PTDA** formats for other versions of OS/2 see:

- PTDA** for OS/2 Warp V4.0 RETAIL kernel
- PTDA** for OS/2 Warp V3.0 ALLSTRICT kernel
- PTDA** for OS/2 Warp V3.0 RETAIL kernel
- PTDA** for OS/2 V2.11 ALLSTRICT kernel
- PTDA** for OS/2 V2.11 RETAIL kernel

Pointers

**TCBpPTDA** points to the PTDA associated with a TCB

**CurrTSD** points to the current TSD.

**pPTDASelf** points to the current PTDA.

Locations

System Arena.

**VM Owner****ptda (0xffcb)****Format**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec'ed)
pPTDANextSibling	+10	4	D	Next sibling's PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling's PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid 'hole'
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process's envt seg
ptda_pBeginLIBPATH	+a8	4	D	
ptda_pEndLIBPATH	+ac	4	D	D75220- support dynamic libpath
ptda_pgpc	+b0	240	S	
ptda_pPVDBPrc	+2f0	4	D	
ptda_pSGSList	+2f4	4	D	
ptda_pexllist	+2f8	4	D	Flat pointer to exit list data
ptda_cdllterm	+2fc	4	D	
CDS_Handle	+300	34	W	array of current directory handles
OEMPtr	+334	2	W	Offset to OEM-Added fields
VerFlg	+336	1	B	Initialize with verify off
LCurDrv	+337	1	B	Logical current drive - Default A:
PCurDrv	+338	1	B	physical drive after assign mapping
LIS_Fgnd	+339	1	B	
FgndOnly	+33a	1	B	foreground only flag
ptda_pad1	+33b	1	B	
ptda_pTCBCritSec	+33c	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+340	4	D	TCBs awaiting CritSec wakeup

ptda_cCritSec	+344	2	W	Critical Section Count
CurrentPDB	+346	2	W	Currently active PDB (V86 segment)
seltss	+348	2	W	
ProcFlag	+34a	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+34c	4	D	pending action bits
ptda_ulExitCode	+350	4	D	Exit code of last task
ptda_ulExitType	+354	4	D	Type of exit
ptda_ulExitTID	+358	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+35c	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+360	2	W	SS relative pointer to a curdir struct
CDSsize	+362	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+364	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+366	2	W	Selector of Profile Control Block
ptda_pPCB	+368	4	D	Pointer to Profile Control Block
JFN_Max	+36c	2	W	highest JFN used so far
NextSrchrH	+36e	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+370	4	D	Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+374	2	W	
FNotifyLocal_Count	+376	2	W	
Sig_ignf	+378	2	W	bit vector of ignored signals
Sig_hndf	+37a	2	W	bit vector of handled signals
Sig_errf	+37c	2	W	bit vector of error generating signals
Sig_attempted	+37e	2	W	bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg	+380	10	W	byte vector of signal arguments
Sig_termtid	+390	2	W	'Terminator' TID for APTERM.
HoldSigCnt	+392	2	W	DOSHOLD SIGNAL counter
SigFocusCnt	+394	2	W	PUBLIB DOS32SET SIGNAL EXCEPTION FOCUS count
JFN_Table	+396	28	W	default handle table
JFN_Flags	+3be	14	B	default JFN flags table
ptda_rasflag	+3d2	2	W	RAS trace indicator
SysSemPTDATbl	+3d4	100	S	
SavedHardErr	+4d4	4	D	
ptda_ptdasem	+4d8	C	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+4e4	C	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+4f0	6	W	current IDT limit/base



Csid	+4f6	2	W	Command Subtree ID
Behav_bit	+4f8	2	W	program behavior bits
MSW	+4fa	2	W	CPU matching status word
ptda_rsrclist	+4fc	4	D	far pointer to local resource list
ptda_pldrdldHead	+500	4	D	loader demand load data list
pPrSemTbl	+504	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+508	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+50c	4	D	number of entries in pPrSemTbl
ulPrNextFree	+510	4	D	next free slot in pPrSemTbl
hksPrTbl	+514	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+518	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+51c	4	D	size of pShSemBmp in bits
hksShBmp	+520	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+524	4	D	number of mutex owned by this process in the two sem tables
ptda_TLMA	+528	4	D	in use flag and dword copy count
ptda_TLMABM	+52c	4	B	thread local memory
ptda_TLMASizeMap	+530	20	B	thread local memory
Cons_Loc	+550	A	S	
SysCallSfcn	+55a	1	B	Value of AL on system entry
SysCall	+55b	1	B	Last system call processed
KBD_Mode	+55c	1	B	Keyboard input mode
ptda_NewFiles	+55d	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+55e	1	B	Non-zero if I 24 FAILED magically
CP_Flgs	+55f	1	B	Default is no codepage in system.
Sig_vec	+560	20	D	signal handlers
Exc_vec	+580	1C	D	OSOLETE exception vectors
ptda_timerhead	+59c	4	D	
ptda_extsig	+5a0	1	B	
ptda_pad6	+5a1	3	B	alignment
pPvwDataBlk	+5a4	4	D	Used by perfview
ptda_lanman_sec	+5a8	4	D	Used by LANMAN & HPFS for security.
SigFTerm	+5ac	2	W	
ptda_ppgdata	+5ae	2	W	offset ptda_pgdata
ptda_child	+5b0	2	W	New child PTDA handle (Child being Exec'ed)
ptda_childalias	+5b2	2	W	
ptda_handle	+5b4	2	W	handle to this segment

ptda_module	+5b6	2	W	program module handle for process
ptda_ldthandle	+5b8	2	W	
ptda_ldtpgmap	+5ba	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+5bc	4	D	
CP_CaseMapTbl	+5c0	4	D	
codepage_tag	+5c4	2	W	the current code page
JFN_Length	+5c6	2	W	Size of JFN table in bytes
JFN_pTable	+5c8	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+5cc	4	D	pointer to JFN flags
ptda_pad	+5d0	1	B	
ExtErr_Locus	+5d1	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+5d2	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+5d4	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+5d5	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+5d6	24	S	
ptda_vme	+5fa	1	B	VME Flag
ptda_pad3	+5fb	1	B	alignment
CurrTCB	+5fc	2	W	pointer to current TCB
CurrTSD	+5fe	2	W	pointer to current TSD
ThisPTDA	+600	2	W	Selector for this ptda
ptda_NPX_em_cs	+602	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+604	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+608	2	W	alignment b726833
ptda_signature	+60a	2	B	must contain "TD"

**ptda\_ForcedActions** flag definitions:

Name	Bit Mask	Description
TK_FF_BUF	0x00000001	Buffer must be released
TK_FF_EXIT	0x00000002	Call TKExit (old FF_DES)
TK_FF_CRITSEC	0x00000004	Enter Per-task critical section
TK_FF_ICE	0x00000008	Freeze thread
TK_FF_NPX	0x00000010	NPX Error
TK_FF_TIB	0x00000020	Update the TIB
TK_FF_TRC	0x00000040	Enter Debug
TK_FF_SIG	0x00000080	Signal pending
TK_FF_CTXH	0x00000100	Pending local context hooks
TK_FF_STIH	0x00000200	Execute STI hooks
TK_FF_VDMBP	0x00000400	Execute VDM BP hooks

TK_FF_RTRY	0x00000800	Retry V86 system call
TK_FF_PIB	0x00001000	Update the PIB
TK_FF_SCH	0x00002000	Do Scheuler Processing
TK_FF_TFBIT	0x00004000	Validate user eflags TF bit
TK_FF_TIBPRI	0x00008000	Update only the priority fields in TIB 59463

-----

## Per-Task Data Area for OS/2 Warp V4.0 RETAIL kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec'ed)
pPTDANextSibling	+10	4	D	Next sibling's PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling's PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid 'hole'
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcdb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process's envt seg
ptda_pBeginLIBPATH	+a8	4	D	
ptda_pEndLIBPATH	+ac	4	D	D75220- support dynamic libpath
ptda_pgpc	+b0	240	S	
ptda_pPVDBProc	+2f0	4	D	
ptda_pSGSList	+2f4	4	D	
ptda_pexllist	+2f8	4	D	Flat pointer to exit list data
ptda_cdllterm	+2fc	4	D	
CDS_Handle	+300	34	W	array of current directory handles

OEMPtr	+334	2	W	Offset to OEM-Added fields
VerFlg	+336	1	B	Initialize with verify off
LCurDrv	+337	1	B	Logical current drive - Default A:
PCurDrv	+338	1	B	physical drive after assign mapping
LIS_Fgnd	+339	1	B	
FgndOnly	+33a	1	B	foreground only flag
ptda_pad1	+33b	1	B	
ptda_pTCBCritSec	+33c	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+340	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+344	2	W	Critical Section Count
CurrentPDB	+346	2	W	Currently active PDB (V86 segment)
seltss	+348	2	W	
ProcFlag	+34a	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+34c	4	D	pending action bits
ptda_ulExitCode	+350	4	D	Exit code of last task
ptda_ulExitType	+354	4	D	Type of exit
ptda_ulExitTID	+358	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+35c	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+360	2	W	SS relative pointer to a curdir struct
CDSsize	+362	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+364	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+366	2	W	Selector of Profile Control Block
ptda_pPCB	+368	4	D	Pointer to Profile Control Block
JFN_Max	+36c	2	W	highest JFN used so far
NextSrchH	+36e	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+370	4	D	Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+374	2	W	
FNotifyLocal_Count	+376	2	W	
Sig_ignf	+378	2	W	bit vector of ignored signals
Sig_hndf	+37a	2	W	bit vector of handled signals
Sig_errf	+37c	2	W	bit vector of error generating signals
Sig_attempted	+37e	2	W	bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg	+380	10	W	byte vector of signal arguments
Sig_termtid	+390	2	W	'Terminator' TID for APTERM.
HoldSigCnt	+392	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+394	2	W	PUBLIB DOS32SETSSIGNALEXCEPTIONFOCUS

				count
JFN_Table	+396	28	W	default handle table
JFN_Flags	+3be	14	B	default JFN flags table
ptda_rasflag	+3d2	2	W	RAS trace indicator
SysSemPTDATbl	+3d4	100	S	
SavedHardErr	+4d4	4	D	
ptda_ptdasem	+4d8	8	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+4e0	8	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+4e8	6	W	current IDT limit/base
Csid	+4ee	2	W	Command Subtree ID
Behav_bit	+4f0	2	W	program behavior bits
MSW	+4f2	2	W	CPU matching status word
ptda_rsrclist	+4f4	4	D	far pointer to local resource list
ptda_pldrdldHead	+4f8	4	D	loader demand load data list
pPrSemTbl	+4fc	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+500	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+504	4	D	number of entries in pPrSemTbl
ulPrNextFree	+508	4	D	next free slot in pPrSemTbl
hksPrTbl	+50c	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+510	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+514	4	D	size of pShSemBmp in bits
hksShBmp	+518	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+51c	4	D	number of mutex owned by this process in the two sem tables
ptda_TLMA	+520	4	D	in use flag and dword copy count
ptda_TLMABM	+524	4	B	thread local memory
ptda_TLMASizeMap	+528	20	B	thread local memory
Cons_Loc	+548	A	S	
SysCallSfcn	+552	1	B	Value of AL on system entry
SysCall	+553	1	B	Last system call processed
KBD_Mode	+554	1	B	Keyboard input mode
ptda_NewFiles	+555	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+556	1	B	Non-zero if I 24 FAILED magically
CP_Flgs	+557	1	B	Default is no codepage in system.
Sig_vec	+558	20	D	signal handlers
Exc_vec	+578	1C	D	OSOLETE exception vectors
ptda_timerhead	+594	4	D	

ptda_extsig	+598	1	B	
ptda_pad6	+59b	3	B	alignment
pPvwDataBlk	+59c	4	D	Used by perfview
ptda_lanman_sec	+5a0	4	D	Used by LANMAN & HPFS for security.
SigFTerm	+5a4	2	W	
ptda_ppgdata	+5a6	2	W	offset ptda_pgdata
ptda_child	+5a8	2	W	New child PTDA handle (Child being Exec'ed)
ptda_childalias	+5aa	2	W	
ptda_handle	+5ac	2	W	handle to this segment
ptda_module	+5ae	2	W	program module handle for process
ptda_ldthandle	+5b0	2	W	
ptda_ldtpgmap	+5b2	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+5b4	4	D	
CP_CaseMapTbl	+5b8	4	D	
codepage_tag	+5bc	2	W	the current code page
JFN_Length	+5be	2	W	Size of JFN table in bytes
JFN_pTable	+5c0	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+5c4	4	D	pointer to JFN flags
ptda_pad	+5c8	1	B	
ExtErr_Locus	+5c9	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+5ca	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+5cc	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+5cd	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+5ce	24	S	
ptda_vme	+5f2	1	B	VME Flag
ptda_pad3	+5f3	1	B	alignment
CurrTCB	+5f4	2	W	pointer to current TCB
CurrTSD	+5f6	2	W	pointer to current TSD
ThisPTDA	+5f8	2	W	Selector for this ptda
ptda_NPX_em_cs	+5fa	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+5fc	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+600	2	W	alignment b726833
ptda_signature	+602	2	B	must contain "TD"

-----

## Per-Task Data Area for OS/2 Warp V3.0 ALLSTRICT kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec'ed)
pPTDANextSibling	+10	4	D	Next sibling's PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling's PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid 'hole'
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process's envt seg
ptda_pBeginLIBPATH	+a8	4	D	
ptda_pEndLIBPATH	+ac	4	D	D75220- support dynamic libpath
ptda_pgpc	+b0	1E0	S	
ptda_pPVDBPrc	+290	4	D	
ptda_pSGSList	+294	4	D	
ptda_pexllist	+298	4	D	Flat pointer to exit list data
ptda_cdllterm	+29c	4	D	
WFP_Start	+2a0	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+2a2	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+2a4	2	W	End of Path component of string.
Curr_Dir_End	+2a6	2	W	
CDS_Handle	+2a8	34	W	*REDIR*
OEMPtr	+2dc	2	W	
LIS_Fgnd	+2de	1	B	
FgndOnly	+2df	1	B	foreground only flag
ptda_pTCBCritSec	+2e0	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+2e4	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+2e8	2	W	Critical Section Count
CurrentPDB	+2ea	2	W	Currently active PDB (V86 segment)

DTAddr	+2ec	4	D	User's I/O transfer address *REDIR*
seltss	+2f0	2	W	
VolID	+2f2	1	B	!0 if vol ID found in dir search
NoSetDir	+2f3	1	B	If TRUE, do not set directory
SpaceFlag	+2f4	1	B	Embedded spaces allowed in FCB
VerFlg	+2f5	1	B	Initialize with verify off
LCurDrv	+2f6	1	B	Logical current drive - Default A:
PCurDrv	+2f7	1	B	physical drive after assign mapping
Creating	+2f8	1	B	
DelAll	+2f9	1	B	
FoundDel	+2fa	1	B	
Found_dev	+2fb	1	B	true => search found a device 3.10
fSplice	+2fc	1	B	true => do a splice in transpath 3.10
ClusFac	+2fd	1	B	sectors/cluster used in dir search
cMeta	+2fe	1	B	components found 3.10
PathNameType	+2ff	1	B	
DevPt	+300	4	D	Address of device found by DevName *REDIR*
DirSec	+304	4	D	
DirStart	+308	2	W	
NxtClusNum	+30a	2	W	
EntFree	+30c	2	W	
EntLast	+30e	2	W	
LastEnt	+310	2	W	
ProcFlag	+312	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+314	4	D	pending action bits
ptda_ulExitCode	+318	4	D	Exit code of last task
ptda_ulExitType	+31c	4	D	Type of exit
ptda_ulExitTID	+320	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+324	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+328	2	W	SS relative pointer to a curdir struct
CDSsize	+32a	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+32c	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+32e	2	W	Selector of Profile Control Block
ptda_pPCB	+330	4	D	Pointer to Profile Control Block
JFN_Max	+334	2	W	highest JFN used so far
NextSrchH	+336	2	W	Next value to use for search handle First value used will be 2.



SrchRmp	+338	4	D	Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+33c	2	W	
FNotifyLocal_Count	+33e	2	W	
Sig_ignf	+340	2	W	bit vector of ignored signals
Sig_hndf	+342	2	W	bit vector of handled signals
Sig_errf	+344	2	W	bit vector of error generating signals
Sig_attempted	+346	2	W	bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg	+348	10	W	byte vector of signal arguments
Sig_termtid	+358	2	W	'Terminator' TID for APTERM.
HoldSigCnt	+35a	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+35c	2	W	PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table	+35e	28	W	default handle table
JFN_Flags	+386	14	B	default JFN flags table
ptda_rasflag	+39a	2	W	RAS trace indicator
SysSemPTDATbl	+39c	100	S	
SavedHardErr	+49c	4	D	
ptda_ptdasem	+4a0	C	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+4ac	C	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+4b8	6	W	current IDT limit/base
Csid	+4be	2	W	Command Subtree ID
Behav_bit	+4c0	2	W	program behavior bits
MSW	+4c2	2	W	CPU matching status word
ptda_rsrclist	+4c4	4	D	far pointer to local resource list
ptda_pldrdldHead	+4c8	4	D	loader demand load data list
pPrSemTbl	+4cc	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+4d0	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+4d4	4	D	number of entries in pPrSemTbl
ulPrNextFree	+4d8	4	D	next free slot in pPrSemTbl
hksPrTbl	+4dc	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+4e0	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+4e4	4	D	size of pShSemBmp in bits
hksShBmp	+4e8	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+4ec	4	D	number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+4f0	2	W	number of share/lock viol retries
RetryCount	+4f2	2	W	num of share/lock retries to do

RetryLoop	+4f4	2	W	num of share/lock retry delay loops ceb 75871
ptda_pSrchrBuf	+4f6	2	W	internal search buffer
ptda_pad1	+4f8	2	W	
ptda_pOpenBuf	+4fa	2	W	
ptda_TLMA	+4fc	4	D	in use flag and dword copy count
ptda_TLMABM	+500	4	B	thread local memory
ptda_TLMASizeMap	+504	20	B	thread local memory
Cons_Loc	+524	A	S	
SysCallSfcn	+52e	1	B	Value of AL on system entry
SysCall	+52f	1	B	Last system call processed
KBD_Mode	+530	1	B	Keyboard input mode
ptda_NewFiles	+531	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+532	1	B	Non-zero if I 24 FAILED magically
ptda_direntry	+533	20	S	
CP_Flgs	+553	1	B	Default is no codepage in system.
Sig_vec	+554	20	D	signal handlers
Exc_vec	+574	1C	D	OSOLETE exception vectors
ptda_timerhead	+590	4	D	
Attrib	+594	2	W	storage for file attributes *REDIR*
ExtFCB	+596	1	B	Extended FCB
ptda_extsig	+597	1	B	
ptda_lanman_sec	+598	4	D	Used by LANMAN & HPFS for security.
ptda_pad2	+59c	2	W	alignment
ptda_ppgdata	+59e	2	W	
ptda_child	+5a0	2	W	New child PTDA handle (Child being Exec'ed)
ptda_childalias	+5a2	2	W	
ptda_handle	+5a4	2	W	handle to this segment
ptda_module	+5a6	2	W	program module handle for process
ptda_ldthandle	+5a8	2	W	
ptda_ldtpgmap	+5aa	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+5ac	4	D	
CP_CaseMapTbl	+5b0	4	D	
codepage_tag	+5b4	2	W	the current code page
JFN_Length	+5b6	2	W	Size of JFN table in bytes
JFN_pTable	+5b8	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+5bc	4	D	pointer to JFN flags
Joins	+5c0	1	B	number of joins
ExtErr_Locus	+5c1	1	B	Extended Error Locus *REDIR* 3.10

ExtErr	+5c2	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+5c4	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+5c5	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+5c6	24	S	
ptda_pad3	+5ea	2	W	alignment
CurrTCB	+5ec	2	W	pointer to current TCB
CurrTSD	+5ee	2	W	pointer to current TSD
ThisPTDA	+5f0	2	W	Selector for this ptda
ptda_NPX_em_cs	+5f2	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+5f4	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+5f8	2	W	alignment b726833
ptda_signature	+5fa	2	B	must contain "TD"

-----

## Per-Task Data Area for OS/2 Warp V3.0 RETAIL kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec'ed)
pPTDANextSibling	+10	4	D	Next sibling's PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling's PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid 'hole'
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcdb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process's envt seg
ptda_pBeginLIBPATH	+a8	4	D	

ptda_pEndLIBPATH	+ac	4	D	D75220- support dynamic libpath
ptda_pgpc	+b0	1E0	S	
ptda_pPVDBPrc	+290	4	D	
ptda_pSGSList	+294	4	D	
ptda_pexllist	+298	4	D	Flat pointer to exit list data
ptda_cdllterm	+29c	4	D	
WFP_Start	+2a0	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+2a2	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+2a4	2	W	End of Path component of string.
Curr_Dir_End	+2a6	2	W	
CDS_Handle	+2a8	34	W	*REDIR*
OEMPtr	+2dc	2	W	
LIS_Fgnd	+2de	1	B	
FgndOnly	+2df	1	B	foreground only flag
ptda_pTCBCritSec	+2e0	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+2e4	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+2e8	2	W	Critical Section Count
CurrentPDB	+2ea	2	W	Currently active PDB (V86 segment)
DTAddr	+2ec	4	D	User's I/O transfer address *REDIR*
seltss	+2f0	2	W	
VolID	+2f2	1	B	!0 if vol ID found in dir search
NoSetDir	+2f3	1	B	If TRUE, do not set directory
SpaceFlag	+2f4	1	B	Embedded spaces allowed in FCB
VerFlg	+2f5	1	B	Initialize with verify off
LCurDrv	+2f6	1	B	Logical current drive - Default A:
PCurDrv	+2f7	1	B	physical drive after assign mapping
Creating	+2f8	1	B	
DelAll	+2f9	1	B	
FoundDel	+2fa	1	B	
Found_dev	+2fb	1	B	true => search found a device 3.10
fSplice	+2fc	1	B	true => do a splice in transpath 3.10
ClusFac	+2fd	1	B	sectors/cluster used in dir search
cMeta	+2fe	1	B	components found 3.10
PathNameType	+2ff	1	B	
DevPt	+300	4	D	Address of device found by DevName *REDIR*
DirSec	+304	4	D	
DirStart	+308	2	W	
NxtClusNum	+30a	2	W	

EntFree	+30c	2	W	
EntLast	+30e	2	W	
LastEnt	+310	2	W	
ProcFlag	+312	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+314	4	D	pending action bits
ptda_ulExitCode	+318	4	D	Exit code of last task
ptda_ulExitType	+31c	4	D	Type of exit
ptda_ulExitTID	+320	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+324	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+328	2	W	SS relative pointer to a curdir struct
CDSsize	+32a	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+32c	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+32e	2	W	Selector of Profile Control Block
ptda_pPCB	+330	4	D	Pointer to Profile Control Block
JFN_Max	+334	2	W	highest JFN used so far
NextSrchH	+336	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+338	4	D	Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+33c	2	W	
FNotifyLocal_Count	+33e	2	W	
Sig_ignf	+340	2	W	bit vector of ignored signals
Sig_hndf	+342	2	W	bit vector of handled signals
Sig_errf	+344	2	W	bit vector of error generating signals
Sig_attempted	+346	2	W	bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg	+348	10	W	byte vector of signal arguments
Sig_termtid	+358	2	W	'Terminator' TID for APTERM.
HoldSigCnt	+35a	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+35c	2	W	PUBLIB DOS32SETSSIGNAL EXCEPTION FOCUS count
JFN_Table	+35e	28	W	default handle table
JFN_Flags	+386	14	B	default JFN flags table
ptda_rasflag	+39a	2	W	RAS trace indicator
SysSemPTDATbl	+39c	100	S	
SavedHardErr	+49c	4	D	
ptda_ptdasem	+4a0	8	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+4a8	8	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+4b0	6	W	current IDT limit/base

Csid	+4b6	2	W	Command Subtree ID
Behav_bit	+4b8	2	W	program behavior bits
MSW	+4ba	2	W	CPU matching status word
ptda_rsrclist	+4bc	4	D	far pointer to local resource list
ptda_pldrdldHead	+4c0	4	D	loader demand load data list
pPrSemTbl	+4c4	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+4c8	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+4cc	4	D	number of entries in pPrSemTbl
ulPrNextFree	+4d0	4	D	next free slot in pPrSemTbl
hksPrTbl	+4d4	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+4d8	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+4dc	4	D	size of pShSemBmp in bits
hksShBmp	+4e0	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+4e4	4	D	number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+4e8	2	W	number of share/lock viol retries
RetryCount	+4ea	2	W	num of share/lock retries to do
RetryLoop	+4ec	2	W	num of share/lock retry delay loops ceb 75871
ptda_pSrchBuf	+4ee	2	W	internal search buffer
ptda_pad1	+4f0	2	W	
ptda_pOpenBuf	+4f2	2	W	
ptda_TLMA	+4f4	4	D	in use flag and dword copy count
ptda_TLMABM	+4f8	4	B	thread local memory
ptda_TLMASizeMap	+4fc	20	B	thread local memory
Cons_Loc	+51c	A	S	
SysCallSfcn	+526	1	B	Value of AL on system entry
SysCall	+527	1	B	Last system call processed
KBD_Mode	+528	1	B	Keyboard input mode
ptda_NewFiles	+529	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+52a	1	B	Non-zero if I 24 FAILED magically
ptda_direntry	+52b	20	S	
CP_Flgs	+54b	1	B	Default is no codepage in system.
Sig_vec	+54c	20	D	signal handlers
Exc_vec	+56c	1C	D	OSOLETE exception vectors
ptda_timerhead	+588	4	D	
Attrib	+58c	2	W	storage for file attributes *REDIR*
ExtFCB	+58e	1	B	Extended FCB

ptda_extsig	+58f	1	B	
ptda_lanman_sec	+590	4	D	Used by LANMAN & HPFS for security.
ptda_pad2	+594	2	W	alignment
ptda_ppgdata	+596	2	W	
ptda_child	+598	2	W	New child PTDA handle (Child being Exec'ed)
ptda_childalias	+59a	2	W	
ptda_handle	+59c	2	W	handle to this segment
ptda_module	+59e	2	W	program module handle for process
ptda_ldthandle	+5a0	2	W	
ptda_ldtpgmap	+5a2	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+5a4	4	D	
CP_CaseMapTbl	+5a8	4	D	
codepage_tag	+5ac	2	W	the current code page
JFN_Length	+5ae	2	W	Size of JFN table in bytes
JFN_pTable	+5b0	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+5b4	4	D	pointer to JFN flags
Joins	+5b8	1	B	number of joins
ExtErr_Locus	+5b9	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+5ba	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+5bc	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+5bd	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+5be	24	S	
ptda_pad3	+5e2	2	W	alignment
CurrTCB	+5e4	2	W	pointer to current TCB
CurrTSD	+5e6	2	W	pointer to current TSD
ThisPTDA	+5e8	2	W	Selector for this ptda
ptda_NPX_em_cs	+5ea	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+5ec	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+5f0	2	W	alignment b726833
ptda_signature	+5f2	2	B	must contain "TD"

-----

## Per-Task Data Area for OS/2 V2.11 ALLSTRICT kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pPTDAParent	+0	4	D	Parent PTDA

pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec'ed)
pPTDANextSibling	+10	4	D	Next sibling's PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling's PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid 'hole'
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process's envt seg
ptda_pgpc	+a8	400	S	
ptda_pmemstatcur	+4a8	4	D	
ptda_memstat	+4ac	3C	S	
ptda_pVDBPrc	+4e8	4	D	
ptda_pSGSList	+4ec	4	D	
ptda_pexllist	+4f0	4	D	Flat pointer to exit list data
ptda_cdllterm	+4f4	4	D	
WFP_Start	+4f8	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+4fa	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+4fc	2	W	End of Path component of string.
Curr_Dir_End	+4fe	2	W	
CDS_Handle	+500	34	W	*REDIR*
OEMPTr	+534	2	W	
LIS_Fgnd	+536	1	B	
FgndOnly	+537	1	B	foreground only flag
ptda_pTCBCritSec	+538	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+53c	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+540	2	W	Critical Section Count
CurrentPDB	+542	2	W	Currently active PDB (V86 segment)
DTAddr	+544	4	D	User's I/O transfer address *REDIR*
seltss	+548	2	W	



VolID	+54a	1	B	!0 if vol ID found in dir search
NoSetDir	+54b	1	B	If TRUE, do not set directory
SpaceFlag	+54c	1	B	Embedded spaces allowed in FCB
VerFlg	+54d	1	B	Initialize with verify off
LCurDrv	+54e	1	B	Logical current drive - Default A:
PCurDrv	+54f	1	B	physical drive after assign mapping
Creating	+550	1	B	
DelAll	+551	1	B	
FoundDel	+552	1	B	
Found_dev	+553	1	B	true => search found a device 3.10
fSplice	+554	1	B	true => do a splice in transpath 3.10
ClusFac	+555	1	B	sectors/cluster used in dir search
cMeta	+556	1	B	components found 3.10
PathNameType	+557	1	B	
DevPt	+558	4	D	Address of device found by DevName *REDIR*
DirSec	+55c	4	D	
DirStart	+560	2	W	
NxtClusNum	+562	2	W	
EntFree	+564	2	W	
EntLast	+566	2	W	
LastEnt	+568	2	W	
ProcFlag	+56a	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.
ptda_ForcedActions	+56c	4	D	pending action bits
ptda_ulExitCode	+570	4	D	Exit code of last task
ptda_ulExitType	+574	4	D	Type of exit
ptda_ulExitTID	+578	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+57c	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+580	2	W	SS relative pointer to a curdir struct
CDSsize	+582	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+584	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+586	2	W	Selector of Profile Control Block
ptda_pPCB	+588	4	D	Pointer to Profile Control Block
JFN_Max	+58c	2	W	highest JFN used so far
NextSrchH	+58e	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+590	4	D	Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+594	2	W	

FNotifyLocal_Count	+596	2	W	
Sig_ignf	+598	2	W	bit vector of ignored signals
Sig_hndf	+59a	2	W	bit vector of handled signals
Sig_errf	+59c	2	W	bit vector of error generating signals
Sig_attempted	+59e	2	W	bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg	+5a0	10	W	byte vector of signal arguments
Sig_termtid	+5b0	2	W	'Terminator' TID for APTERM.
HoldSigCnt	+5b2	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+5b4	2	W	PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table	+5b6	28	W	default handle table
JFN_Flags	+5de	14	B	default JFN flags table
ptda_rasflag	+5f2	2	W	RAS trace indicator
SysSemPTDATbl	+5f4	100	S	
SavedHardErr	+6f4	4	D	
ptda_ptdasem	+6f8	C	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+704	C	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+710	6	W	current IDT limit/base
Csid	+716	2	W	Command Subtree ID
Behav_bit	+718	2	W	program behavior bits
MSW	+71a	2	W	CPU matching status word
ptda_rsrclist	+71c	4	D	far pointer to local resource list
ptda_pldrdldHead	+720	4	D	loader demand load data list
pPrSemTbl	+724	4	D	(void * => PSEM) pointer to private semaphore table
ulPrTblSize	+728	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+72c	4	D	number of entries in pPrSemTbl
ulPrNextFree	+730	4	D	next free slot in pPrSemTbl
hksPrTbl	+734	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+738	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+73c	4	D	size of pShSemBmp in bits
hksShBmp	+740	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+744	4	D	number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+748	2	W	number of share/lock viol retries
RetryCount	+74a	2	W	num of share/lock retries to do
ptda_pad1	+74c	2	W	alignment
ptda_pSrchBuf	+74e	2	W	internal search buffer
ptda_LibiError	+750	2	W	reuse same field to hold library init

errors

ptda_pOpenBuf	+752	2	W	
Cons_Loc	+754	A	S	
SysCallSfcn	+75e	1	B	Value of AL on system entry
SysCall	+75f	1	B	Last system call processed
KBD_Mode	+760	1	B	Keyboard input mode
ptda_NewFiles	+761	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+762	1	B	Non-zero if I 24 FAILED magically
ptda_direntry	+763	20	S	
CP_Flgs	+783	1	B	Default is no codepage in system.
Sig_vec	+784	20	D	signal handlers
Exc_vec	+7a4	1C	D	OSOLETE exception vectors
ptda_timerhead	+7c0	4	D	
Attrib	+7c4	2	W	storage for file attributes *REDIR*
ExtFCB	+7c6	1	B	Extended FCB
ptda_extsig	+7c7	1	B	
ptda_lanman_sec	+7c8	4	D	Used by LANMAN & HPFS for security.
ptda_pad2	+7cc	2	W	alignment
ptda_ppgdata	+7ce	2	W	
ptda_child	+7d0	2	W	New child PTDA handle (Child being Exec'ed)
ptda_childalias	+7d2	2	W	
ptda_handle	+7d4	2	W	handle to this segment
ptda_module	+7d6	2	W	program module handle for process
ptda_ldthandle	+7d8	2	W	
ptda_ldtpgmap	+7da	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+7dc	4	D	
CP_CaseMapTbl	+7e0	4	D	
codepage_tag	+7e4	2	W	the current code page
JFN_Length	+7e6	2	W	Size of JFN table in bytes
JFN_pTable	+7e8	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+7ec	4	D	pointer to JFN flags
Joins	+7f0	1	B	number of joins
ExtErr_Locus	+7f1	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+7f2	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+7f4	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+7f5	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+7f6	24	S	
ptda_pad3	+81a	2	W	alignment
CurrTCB	+81c	2	W	pointer to current TCB

CurrTSD	+81e	2	W	pointer to current TSD
ThisPTDA	+820	2	W	Selector for this ptda
ptda_NPX_em_cs	+822	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+824	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+828	2	W	alignment b726833
ptda_signature	+82a	2	B	must contain "TD"

-----

## Per-Task Data Area for OS/2 V2.11 RETAIL kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pPTDAParent	+0	4	D	Parent PTDA
pPTDASelf	+4	4	D	This PTDA
pPTDAFirstChild	+8	4	D	Head of child chain PTDA
pPTDAExecChild	+c	4	D	New Child PTDA (Child being exec'ed)
pPTDANextSibling	+10	4	D	Next sibling's PTDA
pPTDAPrevSibling	+14	4	D	Previous sibling's PTDA
ptda_pszproc	+18	4	D	Pointer to the EXE file this process is executing. Used by PerfView
ptda_pTCBHole	+1c	4	D	some TCB before first Tid 'hole'
ptda_pTCBHead	+20	4	D	Head of list of active TCBs owned by this process
ptda_cTCB	+24	2	W	Number of TCBs in use
ptda_ctib	+26	2	W	Count of TIBs allocated
ptda_avatib	+28	10	D	Pointers to TIB arrays
ptda_pdcdb	+38	4	D	
ptda_flDbg	+3c	4	D	
ptda_ah	+40	40	S	Private arena header
ptda_pgdata	+80	26	S	
ptda_environ	+a6	2	W	handle to process's envt seg
ptda_pgpc	+a8	400	S	
ptda_pmemstatcur	+4a8	4	D	
ptda_memstat	+4ac	3C	S	
ptda_pPVDBProc	+4e8	4	D	
ptda_pSGSList	+4ec	4	D	
ptda_pexllist	+4f0	4	D	Flat pointer to exit list data
ptda_cdllterm	+4f4	4	D	

WFP_Start	+4f8	2	W	TASKAREA offset for working string *REDIR*
Ren_WFP	+4fa	2	W	WFB pointer for rename destination *REDIR*
WFP_Path_End	+4fc	2	W	End of Path component of string.
Curr_Dir_End	+4fe	2	W	
CDS_Handle	+500	34	W	*REDIR*
OEMPtr	+534	2	W	
LIS_Fgnd	+536	1	B	
FgndOnly	+537	1	B	foreground only flag
ptda_pTCBCritSec	+538	4	D	TCB that did enter CritSec
ptda_pTCBPriQCritSec	+53c	4	D	TCBs awaiting CritSec wakeup
ptda_cCritSec	+540	2	W	Critical Section Count
CurrentPDB	+542	2	W	Currently active PDB (V86 segment)
DTAddr	+544	4	D	User's I/O transfer address *REDIR*
seltss	+548	2	W	
VolID	+54a	1	B	!0 if vol ID found in dir search
NoSetDir	+54b	1	B	If TRUE, do not set directory
SpaceFlag	+54c	1	B	Embedded spaces allowed in FCB
VerFlg	+54d	1	B	Initialize with verify off
LCurDrv	+54e	1	B	Logical current drive - Default A:
PCurDrv	+54f	1	B	physical drive after assign mapping
Creating	+550	1	B	
DelAll	+551	1	B	
FoundDel	+552	1	B	
Found_dev	+553	1	B	true => search found a device 3.10
fSplice	+554	1	B	true => do a splice in transpath 3.10
ClusFac	+555	1	B	sectors/cluster used in dir search
cMeta	+556	1	B	components found 3.10
PathNameType	+557	1	B	
DevPt	+558	4	D	Address of device found by DevName *REDIR*
DirSec	+55c	4	D	
DirStart	+560	2	W	
NxtClusNum	+562	2	W	
EntFree	+564	2	W	
EntLast	+566	2	W	
LastEnt	+568	2	W	
ProcFlag	+56a	2	W	if == 1 then this is a special process (swapper or screen switch); NO removable media buffer will be allocated to this process.

ptda_ForcedActions	+56c	4	D	pending action bits
ptda_ulExitCode	+570	4	D	Exit code of last task
ptda_ulExitType	+574	4	D	Type of exit
ptda_ulExitTID	+578	4	D	Exit Thread ID (32-bit exceptions)
ThisCDS	+57c	4	D	Address of current CDS *REDIR* 3.10
ptda_pCDS	+580	2	W	SS relative pointer to a curdir struct
CDSsize	+582	2	W	Size of CDS pointed to by ThisCDS ONLY used for CDS entries in RMP seg
Sattrib	+584	2	W	Storage for search attrs *REDIR* 3.10
sPCB	+586	2	W	Selector of Profile Control Block
ptda_pPCB	+588	4	D	Pointer to Profile Control Block
JFN_Max	+58c	2	W	highest JFN used so far
NextSrchrH	+58e	2	W	Next value to use for search handle First value used will be 2.
SrchRmp	+590	4	D	Handle & Selector for RMP segment we keep search handles in.
FNotifyLocal_First	+594	2	W	
FNotifyLocal_Count	+596	2	W	
Sig_ignf	+598	2	W	bit vector of ignored signals
Sig_hndf	+59a	2	W	bit vector of handled signals
Sig_errf	+59c	2	W	bit vector of error generating signals
Sig_attempted	+59e	2	W	bit vector of signals we've tried to handle with 32-bit exceptions
Sig_arg	+5a0	10	W	byte vector of signal arguments
Sig_termtid	+5b0	2	W	'Terminator' TID for APTERM.
HoldSigCnt	+5b2	2	W	DOSHOLDSIGNAL counter
SigFocusCnt	+5b4	2	W	PUBLIB DOS32SETSIGNALEXCEPTIONFOCUS count
JFN_Table	+5b6	28	W	default handle table
JFN_Flags	+5de	14	B	default JFN flags table
ptda_rasflag	+5f2	2	W	RAS trace indicator
SysSemPTDATbl	+5f4	100	S	
SavedHardErr	+6f4	4	D	
ptda_ptdasem	+6f8	8	S	PTDA semaphore that is, inter-thread
ptda_DLMsem	+700	8	S	b732954 Edd PTDA semaphore that is, inter-thread
ptda_lidt	+708	6	W	current IDT limit/base
Csid	+70e	2	W	Command Subtree ID
Behav_bit	+710	2	W	program behavior bits
MSW	+712	2	W	CPU matching status word
ptda_rsrclist	+714	4	D	far pointer to local resource list
ptda_pldrldHead	+718	4	D	loader demand load data list
pPrSemTbl	+71c	4	D	(void * => PSEM) pointer to private

semaphore table				
ulPrTblSize	+720	4	D	size of pPrSemTbl in dwords
ulPrTotUsed	+724	4	D	number of entries in pPrSemTbl
ulPrNextFree	+728	4	D	next free slot in pPrSemTbl
hksPrTbl	+72c	4	D	kernel semaphore handle for private semaphore table
pShSemBmp	+730	4	D	pointer to private bitmap for the shared semaphore table
ulShBmpSize	+734	4	D	size of pShSemBmp in bits
hksShBmp	+738	4	D	kernel semaphore handle for private semaphore table
ulMtxOwned	+73c	4	D	number of mutex owned by this process in the two sem tables
ShareRetriesLeft	+740	2	W	number of share/lock viol retries
RetryCount	+742	2	W	num of share/lock retries to do
RetryLoop	+744	2	W	
ptda_pSrchBuf	+746	2	W	internal search buffer
ptda_LibiError	+748	2	W	reuse same field to hold library init errors
ptda_pOpenBuf	+74a	2	W	
Cons_Loc	+74c	A	S	
SysCallSfcn	+756	1	B	Value of AL on system entry
SysCall	+757	1	B	Last system call processed
KBD_Mode	+758	1	B	Keyboard input mode
ptda_NewFiles	+759	1	B	If bit one is set, process supports // 54400 new files (long names)
AutoFail	+75a	1	B	Non-zero if I 24 FAILED magically
ptda_direntry	+75b	20	S	
CP_Flgs	+77b	1	B	Default is no codepage in system.
Sig_vec	+77c	20	D	signal handlers
Exc_vec	+79c	1C	D	OSOLETE exception vectors
ptda_timerhead	+7b8	4	D	
Attrib	+7bc	2	W	storage for file attributes *REDIR*
ExtFCB	+7be	1	B	Extended FCB
ptda_extsig	+7bf	1	B	
ptda_lanman_sec	+7c0	4	D	Used by LANMAN & HPFS for security.
ptda_pad2	+7c4	2	W	alignment
ptda_ppgdata	+7c6	2	W	
ptda_child	+7c8	2	W	New child PTDA handle (Child being Exec'ed)
ptda_childalias	+7ca	2	W	
ptda_handle	+7cc	2	W	handle to this segment
ptda_module	+7ce	2	W	program module handle for process

ptda_ldthandle	+7d0	2	W	
ptda_ldtpgmap	+7d2	2	W	Bitmap of valid LDT pages
ptda_ldtaddr	+7d4	4	D	
CP_CaseMapTbl	+7d8	4	D	
codepage_tag	+7dc	2	W	the current code page
JFN_Length	+7de	2	W	Size of JFN table in bytes
JFN_pTable	+7e0	4	D	PM pointer to JFN table
JFN_Flg_Ptr	+7e4	4	D	pointer to JFN flags
Joins	+7e8	1	B	number of joins
ExtErr_Locus	+7e9	1	B	Extended Error Locus *REDIR* 3.10
ExtErr	+7ea	2	W	Extended Error code *REDIR* 3.10
ExtErr_Action	+7ec	1	B	Extended Error Action *REDIR* 3.10
ExtErr_Class	+7ed	1	B	Extended Error Class *REDIR* 3.10
ptda_infoseg	+7ee	24	S	
ptda_pad3	+812	2	W	alignment
CurrTCB	+814	2	W	pointer to current TCB
CurrTSD	+816	2	W	pointer to current TSD
ThisPTDA	+818	2	W	Selector for this ptda
ptda_NPX_em_cs	+81a	2	W	b726833 NPX emulator CS b726833
ptda_NPX_em_eip	+81c	4	D	b726833 NPX emulator EIP b726833
ptda_pad4	+820	2	W	alignment b726833
ptda_signature	+822	2	B	must contain "TD"

-----

## Local Information Segement

### Pointers

SAS field **SAS\_info\_local** points to the current LISEG.

### Locations

**dfff:0** is the address of the copy of the LISEG for the current thread & process.

The LISEG for each process is imbedded in the PTDA at **ptda\_infoseg**.

### VM Owner

**infoseg (0xff75)**

### Format

#### InfoSegLDT

Field Name	Offset	Length	Type	Description
LIS_CurProcID	+0	2	W	Current process ID



LIS_ParProcID	+2	2	W	Process ID of parent
LIS_CurThrdPri	+4	2	W	Current thread priority
LIS_CurThrdID	+6	2	W	Current thread ID
LIS_CurScrnGrp	+8	2	W	Screen group
LIS_ProcStatus	+a	1	B	Process status bits
LIS_fillbyte1	+b	1	B	filler byte
LIS_Fgnd	+c	2	W	Current process is in foreground
LIS_ProcType	+e	1	B	Current process type
LIS_fillbyte2	+f	1	B	filler byte
LIS_AX	+10	2	W	@@V1 Environment selector
LIS_BX	+12	2	W	@@V1 Offset of command line start
LIS_CX	+14	2	W	@@V1 Length of Data Segment
LIS_DX	+16	2	W	@@V1 STACKSIZE from the .EXE file
LIS_SI	+18	2	W	@@V1 HEAPSIZE from the .EXE file
LIS_DI	+1a	2	W	@@V1 Module handle of the application
LIS_DS	+1c	2	W	@@V1 Data Segment Handle of application
LIS_PackSel	+1e	2	W	First tiled selector in this EXE
LIS_PackShrSel	+20	2	W	First selector above shared arena
LIS_PackPckSel	+22	2	W	First selector above packed arena

#### LIS\_ProcStatus flag definitions:

Name	Bit Mask	Description
PS_XITLST	0x01	Doing ExitList Processing
PS_XITTH1	0x02	Exiting thread 1
PS_XITALL	0x04	The whole process is exiting
PS_SYNCARENT	0x10	Parent cares about termination
PS_WAITPARENT	0x20	Parent did an exec-and-wait
PS_DYING	0x40	Process is dying
PS_EMBRYO	0x80	Process in embryonic state

#### LIS\_ProcType flag definitions:

Name	Value	Description
LIS_PT_FULLSCREEN	0	Full screen app.
LIS_PT_REALMODE	1	Real mode process
PT_VDM	1	VDM
LIS_PT_VIOWIN	2	VIO windowable app.
LIS_PT_PRESMGR	3	Presentation Manager app.
LIS_PT_DETACHED	4	Detached app.

# Global Information Segement

## Pointers

SAS field **SAS\_info\_global** points to the current GISEG.

## Locations

**dff4:0** is the address of the copy of the GISEG for the current thread & process.

## VM Owner

**infoseg (0xff75)** - shared arena copy

**os2krnl (0xffaa)** - system arena copy

## Format

### InfoSegGDT

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
SIS_BigTime	+0	4	D	Time from 1-1-1970 in seconds
SIS_MsCount	+4	4	D	Freerunning milliseconds counter
SIS_HrsTime	+8	1	B	Hours
SIS_MinTime	+9	1	B	Minutes
SIS_SecTime	+a	1	B	Seconds
SIS_HunTime	+b	1	B	Hundredths of seconds
SIS_TimeZone	+c	2	W	Timezone in min from GMT (Set to EST)
SIS_ClkIntrvl	+e	2	W	Timer interval (units=0.0001 secs)
SIS_DayDate	+10	1	B	Day-of-month (1-31)
SIS_MonDate	+11	1	B	Month (1-12)
SIS_YrsDate	+12	2	W	Year (>= 1980)
SIS_DOWDate	+14	1	B	Day-of-week (1-1-80 = Tues = 3)
SIS_VerMajor	+15	1	B	Major version number
SIS_VerMinor	+16	1	B	Minor version number
SIS_RevLettr	+17	1	B	Revision letter
SIS_CurScrnGrp	+18	1	B	Fgnd screen group #
SIS_MaxScrnGrp	+19	1	B	Maximum number of screen groups
SIS_HugeShfCnt	+1a	1	B	Shift count for huge segments
SIS_ProtMdOnly	+1b	1	B	Protect-mode-only indicator
SIS_FgndPID	+1c	2	W	Foreground process ID
SIS_Dynamic	+1e	1	B	Dynamic variation flag (1=enabled)
SIS_MaxWait	+1f	1	B	Maxwait (seconds)
SIS_MinSlice	+20	2	W	Minimum timeslice (milliseconds)

SIS_MaxSlice	+22	2	W	Maximum timeslice (milliseconds)
SIS_BootDrv	+24	2	W	Drive from which system was booted
SIS_mec_table	+26	20	B	Table of RAS Major Event Codes (MECs)
SIS_MaxVioWinSG	+46	1	B	Max. no. of VIO windowable SG's
SIS_MaxPresMgrSG	+47	1	B	Max. no. of Presentation Manager SG's
SIS_SysLog	+48	2	W	Error Logging Status
SIS_MMIOBase	+4a	2	W	Memory mapped I/O selector
SIS_MMIOAddr	+4c	4	D	Memory mapped I/O address
SIS_MaxVDMs	+50	1	B	Max. no. of Virtual DOS machines
SIS_Reserved	+51	1	B	

#### SIS\_SysLog flag definitions:

Name	Bit	Mask	Description
LF_LOGENABLE	0x0001		Logging enabled
LF_LOGAVAILABLE	0x0002		Logging available

-----

## Process Information Block

#### Pointers

PTDA field **ptda\_avatib** points to the PIB for the related process.

PIB field **pib\_pchenv** points to the process' environment strings.

#### Locations

Allocated in the Process' Private Arena.

#### VM Owner

PIB owner id: **tktib (0xff3f)** (also used for TIB ownership).

Environment Owner id: **tkenv (0xff3e)**.

#### Format

**PIB** Process Information Block.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pib_ulpid	+0	4	D	Process I.D.
pib_ulppid	+4	4	D	Parent process I.D.
pib_hmte	+8	4	D	Program (.EXE) module handle
pib_pchcmd	+c	2	W	Command line pointer
pib_pchenv	+10	4	D	Environment pointer
pib_flstatus	+14	4	D	Process' status bits
pib_ultype	+18	4	D	Process' type code

**plib\_flgstatus** flag definitions:

Name	Bit Mask	Description
PS_XITLST	0x01	Doing ExitList Processing
PS_XITTH1	0x02	Exiting thread 1
PS_XITALL	0x04	The whole process is exiting
PS_SYNCPARENT	0x10	Parent cares about termination
PS_WAITPARENT	0x20	Parent did an exec-and-wait
PS_DYING	0x40	Process is dying
PS_EMBRYO	0x80	Process in embryonic state

**plib\_ultype** flag definitions:

Name	Value	Description
LIS_PT_FULLSCRN	0	Full screen app.
LIS_PT_REALMODE	1	Real mode process
PT_VDM	1	VDM
LIS_PT_VIOWIN	2	VIO windowable app.
LIS_PT_PRESMGR	3	Presentation Manager app.
LIS_PT_DETACHED	4	Detached app.

-----

## Thread Information Block

**Pointers**

- TCB field **TCBptib** points to the TIB for the related thread.
- TIB field **tib\_ptib2** points to the associated TIB2.
- GDT Selector 150b maps the TIB and is the default value for the FS register.

**Locations**

- Allocated in the Process' Private Arena.

**VM Owner**

- tktib (0xff3f)** (also used for PIB ownership).

**Format**

**TIB** Thread Information Block system independent section.

Field Name	Offset	Length	Type	Description
tib_pexchain	+0	4	D	Head of exception handler chain

tib_pstack	+4	4	D	Pointer to base of stack
tib_pstacklimit	+8	4	D	Pointer to end of stack
tib_ptib2	+c	2	W	Pointer to system specific TIB
tib_version	+10	4	D	Version number for this TIB structure
tib_ordinal	+14	4	D	Thread Ordinal Number

**TIB2** Thread Information Block system dependent section.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
tib2_ultid	+0	4	D	Thread I.D.
tib2_ulpri	+4	4	D	Thread priority
tib2_version	+8	4	D	Version number for this structure
tib2_usMCCount	+c	2	W	Must Complete count
tib2_fMCForceFlag	+e	2	W	Must Complete force flag

## System Stack Frames Client Register Information

### Pointers

TCB field **TCB\_pcriFrameType** points to the CRI.

### Locations

**\_criISF** locates the Interrupt Stack Frame CRI.

**\_criTSF** locates the Trap Stack Frame CRI.

**\_criVSF** locates the VDM Stack Frame CRI.

**\_criSEF** locates the System Entry Stack Frame CRI.

**\_criPASCALSEF** locates the PASCAL System Entry Stack Frame CRI.

**\_criSSF** locates the SCI Stack Frame CRI.

**\_criDHF** locates the Device Help Stack Frame CRI.

**fpoldstack** contains a 32-bit far pointer to the ISF built by the Interrupt Router at interrupt time.

### VM Owner

**os2krnl (0xffaa)**

### Format

### CRI Client Register Information

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
cri_ulSize	+0	4	D	size of stack frame
cri_eax	+4	4	S	eax rip
cri_ebx	+8	4	S	ebx rip

cri_ecx	+c	4	S	ecx rip
cri_edx	+10	4	S	edx rip
cri_ebp	+14	4	S	ebp rip
cri_esi	+18	4	S	esi rip
cri_edi	+1c	4	S	edi rip
cri_ds	+20	4	S	ds rip
cri_es	+24	4	S	es rip
cri_fs	+28	4	S	fs rip
cri_gs	+2c	4	S	gs rip
cri_cs	+30	4	S	cs rip
cri_eip	+34	4	S	eip rip
cri_eflag	+38	4	S	eflag rip
cri_ss	+3c	4	S	ss rip
cri_esp	+40	4	S	esp rip
cri_cbargs	+44	4	S	cbargs rip
cri_trapnum	+48	4	S	trapnum rip
cri_errcode	+4c	4	S	errcode rip
cri_pfnRebuild	+50	4	D	
cri_pfpfnKernelExit	+54	4	D	

#### RIP Register Information Packet

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
rip_flags	+0	2	W	Flags
rip_offset	+2	4	W	Offset of register into stack frame

#### rip\_flags flag definitions:

Name	Bit Mask	Description
KM_RIP_INVALID	0x0001	invalid register
KM_RIP_INVALID_SET	0x0002	invalid register to set
KM_RIP_WORD	0x0004	word register
KM_RIP_TSD_RELATIVE	0x0008	rip_offset relative to TSD beginning
KM_RIP_C32	0x0010	32-bit C style call

#### ISF Interrupt Manager Stack Frame

This is what the stack frame looks like when the system is entered thru the interrupt manager during a hardware interrupt. For a hardware interrupt in a VDM context, the stack frame always needs to be a "VSF" type so the stack frame base is adjusted by ISF\_VSF\_START. The points the stack frame base to "isf\_edi" in the regular interrupt frame. The interrupt stack frame has also been padded (ISF\_STACK\_PAD) between the general registers (EDI to EAX) and the hardware pushed registers (EIP to SS) with a dummy trap number and error code to look like the VSF stack frame.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
isf_CurrIntLevel	+0	4	D	
isf_gs	+4	2	W	
isf_padgs	+6	2	W	
isf_fs	+8	2	W	
isf_padfs	+a	2	W	
isf_es	+c	2	W	
isf_pades	+e	2	W	
isf_ds	+10	2	W	
isf_padds	+12	2	W	
isf_edl	+14	4	D	start of VDM stack frame
isf_esi	+18	4	D	
isf_ebp	+1c	4	D	
isf_padesp	+20	4	D	
isf_ebx	+24	4	D	
isf_edx	+28	4	D	
isf_ecx	+2c	4	D	
isf_eax	+30	4	D	
isf_pad	+34	8	B	
isf_eip	+3c	4	D	
isf_cs	+40	2	W	
isf_padcs	+42	2	W	
isf_eflag	+44	4	D	
isf_esp	+48	4	D	
isf_ss	+4c	2	W	
isf_padss	+4e	2	W	

#### **TSF Trap or Exception Stack Frame**

This is what the stack frame looks like when the system is entered thru a 386 exception (from protected mode).

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
tsf_edl	+0	4	D	
tsf_esi	+4	4	D	
tsf_ebp	+8	4	D	
tsf_padesp	+c	4	D	
tsf_ebx	+10	4	D	
tsf_edx	+14	4	D	
tsf_ecx	+18	4	D	
tsf_eax	+1c	4	D	
tsf_gs	+20	2	W	

tsf_padgs	+22	2	W
tsf_fs	+24	2	W
tsf_padfs	+26	2	W
tsf_es	+28	2	W
tsf_pades	+2a	2	W
tsf_ds	+2c	2	W
tsf_padds	+2e	2	W
tsf_trapnum	+30	4	D
tsf_errcode	+34	4	D
tsf_eip	+38	4	D
tsf_cs	+3c	2	W
tsf_padcs	+3e	2	W
tsf_eflag	+40	4	D
tsf_esp	+44	4	D
tsf_ss	+48	2	W
tsf_padss	+4a	2	W

#### **KSF** Kernel Stack Frame

This is what the stack frame looks like when the system is re-entered from ring 0. This is frame used for handling exception while already in kernel mode.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ksf_edi	+0	4	D	
ksf_esi	+4	4	D	
ksf_ebp	+8	4	D	
ksf_padesp	+c	4	D	
ksf_ebx	+10	4	D	
ksf_edx	+14	4	D	
ksf_ecx	+18	4	D	
ksf_eax	+1c	4	D	
ksf_gs	+20	2	W	
ksf_padgs	+22	2	W	
ksf_fs	+24	2	W	
ksf_padfs	+26	2	W	
ksf_es	+28	2	W	
ksf_pades	+2a	2	W	
ksf_ds	+2c	2	W	
ksf_padds	+2e	2	W	
ksf_trapnum	+30	4	D	
ksf_errcode	+34	4	D	



ksf_eip	+38	4	D
ksf_cs	+3c	2	W
ksf_padcs	+3e	2	W
ksf_eflag	+40	4	D

#### VSF VDM Process Stack Frame

This is what the stack frame looks like when the system is entered from a VDM thru a exception, software or hardware interrupt. Most of the 8086 emulation code uses this stack frame directly for performance. For hardware interrupts taken in a VDM (in either V86 mode or protected mode), the interrupt stack frame (see ISF) is adjusted to look like this frame.

The alternate stack frame holds the real or protected mode sensitive registers for the other mode. So when the VDM is in protected mode, the last V86 mode segment registers and CS:EIP, SS:ESP can be accessed, etc. Two things happen with we mode switch: 1) the alternate register set is exchanged with the regular set (vsf\_eip to vsf\_padgs is the exchanged with vsf\_alteip to vsf\_altpadgs), 2) the TSS's ESP0 value is changed to the appropriate place in the VSF structure. For V86 mode, ESP0 points to the begining of the segment registers (vsf\_gs/vsf\_padgs) and for protected mode ESP0 points to the SS register (vsf\_ss/vsf\_padss). For protected mode entry, the segments registers are stored in vsf\_ds to vsf\_gs explicitly. This makes the V86 mode and protected mode stack frames the same for VDDs and the MVDM kernel code.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
vsf_edi	+0	4	D	
vsf_esi	+4	4	D	
vsf_ebp	+8	4	D	
vsf_padesp	+c	4	D	
vsf_ebx	+10	4	D	
vsf_edx	+14	4	D	
vsf_ecx	+18	4	D	
vsf_eax	+1c	4	D	
vsf_trapnum	+20	4	D	
vsf_errrcode	+24	4	D	
vsf_eip	+28	4	D	
vsf_cs	+2c	2	W	
vsf_padcs	+2e	2	W	
vsf_eflag	+30	4	D	
vsf_esp	+34	4	D	
vsf_ss	+38	2	W	
vsf_padss	+3a	2	W	
vsf_es	+3c	2	W	
vsf_pades	+3e	2	W	
vsf_ds	+40	2	W	
vsf_padds	+42	2	W	
vsf_fs	+44	2	W	
vsf_padfs	+46	2	W	
vsf_gs	+48	2	W	
vsf_padgs	+4a	2	W	

vsf_alteip	+4c	4	D
vsf_altcs	+50	2	W
vsf_altpadcs	+52	2	W
vsf_alteflag	+54	4	D
vsf_altesp	+58	4	D
vsf_altss	+5c	2	W
vsf_altpadss	+5e	2	W
vsf_altes	+60	2	W
vsf_altpades	+62	2	W
vsf_altds	+64	2	W
vsf_altpadds	+66	2	W
vsf_altfs	+68	2	W
vsf_altpadfs	+6a	2	W
vsf_altgs	+6c	2	W
vsf_altpadgs	+6e	2	W

### SEF System Entry Stack Frame

This is the frame put on the by the call gate system entry function (KMEnterKmodeCallGate or KMEnterKmodeAPI32).

This frame is used for:

- 32-bit C APIs, with C callable workers (criSEF)
- 16-bit PASCAL APIs, with C callable workers (criPASCALSEF)

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
sef_edi	+0	4	D	
sef_esi	+4	4	D	
sef_ebp	+8	4	D	
sef_padesp	+c	4	D	
sef_ebx	+10	4	D	
sef_edx	+14	4	D	
sef_ecx	+18	4	D	
sef_eax	+1c	4	D	
sef_gs	+20	2	W	
sef_padgs	+22	2	W	
sef_fs	+24	2	W	
sef_padfs	+26	2	W	
sef_es	+28	2	W	
sef_pades	+2a	2	W	
sef_ds	+2c	2	W	
sef_padds	+2e	2	W	

sef_retaddr	+30	4	D
sef_cbargs	+34	4	D
sef_eflag	+38	4	D
sef_eip	+3c	4	D
sef_cs	+40	2	W
sef_padcs	+42	2	W

#### SCI System Call Interpreter Call Gate Stack Frame

This is what the stack frame looks like when the system is entered thru SCI via call gate using the KMEnterKmodeSCI function.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ssf_edi	+0	4	D	
ssf_esi	+4	4	D	
ssf_ebp	+8	4	D	
ssf_padesp	+c	4	D	
ssf_ebx	+10	4	D	
ssf_edx	+14	4	D	
ssf_ecx	+18	4	D	
ssf_eax	+1c	4	D	
ssf_gs	+20	2	W	
ssf_padgs	+22	2	W	
ssf_fs	+24	2	W	
ssf_padfs	+26	2	W	
ssf_es	+28	2	W	
ssf_pades	+2a	2	W	
ssf_ds	+2c	2	W	
ssf_padds	+2e	2	W	
ssf_thopadr	+30	4	D	
ssf_cbargs	+34	4	D	The Most Significant Bit of cbargs in an SCI stack frame is used to denote that a 16 bit callgate is being used with the SCI mechanism. Used by Dynamic API's.
ssf_sciret	+38	2	W	
ssf_eflag	+3a	4	D	
ssf_eip	+3e	4	D	
ssf_cs	+42	2	W	
ssf_padcs	+42	2	W	

#### DHF Device Help Stack Frame

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	---------------	---------------	-------------	--------------------

dhf_edi	+0	4	D
dhf_esi	+4	4	D
dhf_ebp	+8	4	D
dhf_padesp	+c	4	D
dhf_ebx	+10	4	D
dhf_edx	+14	4	D
dhf_ecx	+18	4	D
dhf_eax	+1c	4	D
dhf_gs	+20	2	W
dhf_padgs	+22	2	W
dhf_fs	+24	2	W
dhf_padfs	+26	2	W
dhf_es	+28	2	W
dhf_pades	+2a	2	W
dhf_ds	+2c	2	W
dhf_padds	+2e	2	W
dhf_eflag	+30	4	D
dhf_scratch	+34	4	D
dhf_eip	+38	4	D
dhf_cs	+3c	2	W
dhf_padcs	+3e	4	D

#### TF Hardware Exception Trap Stack Frame

Stack frame for the trap manager before we go into kernel mode.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
tf_trapnum	+0	4	D	
tf_errcode	+4	4	D	
tf_eip	+8	4	D	
tf_cs	+c	2	W	
tf_padcs	+e	2	W	
tf_eflags	+10	4	D	
tf_esp	+14	4	D	
tf_ss	+18	2	W	
tf_padss	+1a	2	W	

-----

## Exit List Entry Data Structure

**Pointers**  
PTDA field **ptda\_pexllist** points to the head of Exit List chain.

**Locations**  
Allocated from the kernel heaps.

**VM Owner**  
**tkextlst (0xffc7)**

**Format**

**EXENT**

Field Name	Offset	Length	Type	Description
exl_next	+0	4	D	link to next block/order
exl_addr	+4	4	D	Exit list routine address
exl_class	+8	2	W	order & position 0 thru 0xFF
exl_type	+a	2	W	16:16 or 0:32

**exl\_type** values:

Name	Value	Description
TK_TYPE16	0	
TK_TYPE32	1	
TK_TYPEDT	2	

-----

## Exception Handler Structures

**Pointers**  
TIB field **tib\_pexchain** points to the head of the chain of EXCEPTIONREGISTRATIONRECORDs.  
The 1st parameter to the exception handler points to the EXCEPTIONREPORTRECORD.  
The 2nd parameter to the exception handler points to the EXCEPTIONREGISTRATIONRECORD.  
The 3rd parameter to the exception handler points to the CONTEXTRECORD.

**Locations**  
Allocated in the Process' Private Arena,.

**VM Owner**  
**tktib (0xff3f)** (also used for PIB ownership).

**Format**

**CONTEXTRECORD** Exception handler context record.

Field Name	Offset	Length	Type	Description
------------	--------	--------	------	-------------

ContextFlags	+0	4	D	Flags
	+4	6c	S	Floating point section
ctx_env	+4	1c	D	Floating point environment
ctx_stack	+20	50	S	Floating point register stack (8 FPREG structures)
	+70	10	S	Segment Register section
ctx_SegGs	+70	4	D	GS segment register
ctx_SegFs	+74	4	D	FS segment register
ctx_SegEs	+78	4	D	ES segment register
ctx_SegDs	+7c	4	D	DS segment register
	+80	18	S	Integer Register section
ctx_RegEdi	+80	4	D	EDI register
ctx_RegEsi	+84	4	D	ESI register
ctx_RegEax	+88	4	D	EAX register
ctx_RegEbx	+8c	4	D	EBX register
ctx_RegEcx	+90	4	D	ECX register
ctx_RegEdx	+94	4	D	EDX register
	+98	18	S	Control Register section
ctx_RegEbp	+98	4	D	EBP register
ctx_RegEip	+9c	4	D	EIP register
ctx_SegCs	+a0	4	D	CS selector
ctx_EFlags	+a4	4	D	Processor Flags register
ctx_RegEsp	+a8	4	D	ESP register
ctx_SegSs	+ac	4	D	SS segment register

#### FPREG Floating Point Register Stack Element

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
losig	+0	4	D	Low significance double-word
hisig	+4	4	D	High significance double-word
signexp	+8	2	W	Exponent

#### ContextFlags flag definitions:

Name	Bit Mask	Description
CONTEXT_CONTROL	0x00000001L	SS:ESP, CS:EIP, EFLAGS, EBP

CONTEXT_INTEGER	0x00000002L	EAX, EBX, ECX, EDX, ESI, EDI
CONTEXT_SEGMENTS	0x00000004L	DS, ES, FS, GS
CONTEXT_FLOATING_POINT	0x00000008L	Numeric coprocessor state

#### EXCEPTIONREPORTRECORD Exception Handler Report Record.

Field Name	Offset	Length	Type	Description
ExceptionNum	+0	4	D	Exception number
fHandlerFlags	+4	4	D	Exception attributes
NestedExceptionReportRecord	+8	4	D	Preceding exception's report record if nested exception
ExceptionAddress	+c	4	D	Exception address
cParameters	+10	4	D	Size of exception specific information
ExceptionInfo	+14	10	D	Exception specific information

#### fHandlerFlags flag definitions:

Name	Bit Mask	Description
EH_NONCONTINUABLE	0x1	Noncontinuable exception
EH_UNWINDING	0x2	Unwind is in progress
EH_EXIT_UNWIND	0x4	Exit unwind is in progress
EH_STACK_INVALID	0x8	Stack out of limits or unaligned
EH_NESTED_CALL	0x10	Nested exception handler call

#### EXCEPTIONREGISTRATIONRECORD Exception Handler Registration Record.

Field Name	Offset	Length	Type	Description
prev_structure	+0	4	D	Previously registered exception handler
ExceptionHandler	+4	4	D	Exception handler entry point address or -1 if end of chain

-----

## Loader Control Block Reference

The following control blocks are described in this section:

[Module Table Entry \(MTE\)](#)

[Swappable Module Table Entry \(SMTE\)](#)

[Object Table Entry \(OTE\)](#)

[Segment Table Entry \(STE\)](#)

# Module Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0

For **MTE** formats for other versions of OS/2 see:

**MTE** for OS/2 V2.11

**Pointers**

- `_mte_h` points to the head of the chain of MTEs.
- `_global_h` points to head of the chain of library module MTEs.

**Locations**

- Dynamically allocated from the kernel resident heap except for the two MTEs that represent kernel interfaces.
- `_DosMosMte` locates the MTE in OS2KRNL for DOSCALLS.DLL.
- `_VDDModMte` locates the MTE in OS2KRNL for MVDM.DLL.

**VM Owner**

- Dynamically allocated MTEs have owner id **ldrmte (0xffa6)**

**Format**

Field Name	Offset	Length	Type	Description
mte_flags2	+0	2	W	Module flags 2
mte_handle	+2	2	W	the handle for this mte
mte_swapmte	+4	4	D	link to swappable mte
mte_link	+8	4	D	link to next mte
mte_flags1	+c	4	D	Module flags 1
mte_impmodcnt	+10	4	D	Num of entries in Imp Mod Name Tbl
mte_sfn	+14	2	W	file system number for open file
mte_usecnt	+16	2	W	.EXE only - use count
mte_modname	+18	8	B	resident module name (zero extended)

**mte\_flags1** flag definitions:

Name	Bit Mask	Description
NOAUTODS	0x00000000	No Auto DS exists
SOLO	0x00000001	Auto DS is shared
INSTANCEDS	0x00000002	Auto DS is not shared
INSTLIBINIT	0x00000004	Per-instance Libinit
GINISETUP	0x00000008	Global Init has been setup
NOINTERNFIXUPS	0x00000010	internal fixups in .EXE-.DLL applied
NOEXTERNFIXUPS	0x00000020	external fixups in .EXE-.DLL applied
CLASS_PROGRAM	0x00000040	Program class



CLASS_GLOBAL	0x00000080	Global class
CLASS_SPECIFIC	0x000000C0	Specific class, as against global
CLASS_ALL	0x00000000	nonspecific class - all modules
CLASS_MASK	0x000000C0	
MTEPROCESSED	0x00000100	MTE being loaded
USED	0x00000200	MTE is referenced - see ldrgc.c
DOSLIB	0x00000400	set if DOSCALL1
DOSMOD	0x00000800	set if DOSCALLS
MTE_MEDIAFIXED	0x00001000	File Media permits discarding
LDRINVALID	0x00002000	module not loadable
PROGRAMMOD	0x00000000	program module
DEVDRVMOD	0x00004000	device driver module
LIBRARYMOD	0x00008000	DLL module
VDDMOD	0x00010000	VDD module
MVDMMOD	0x00020000	Set if VDD Helper MTE (MVDM.DLL)
INGRAPH	0x00040000	In Module Graph - see ldrgc.c
GINIDONE	0x00080000	Global Init has finished
MTEADDRALLOCED	0x00100000	Allocate specific or not
FSDMOD	0x00200000	FSD MTE
FSHMOD	0x00400000	FS helper MTE
MTELONGNAMES	0x00800000	Module supports long-names
MTE_MEDIACONTIG	0x01000000	File Media contiguous memory req
MTE_MEDIA16M	0x02000000	File Media requires mem below 16M
MTESWAPONLOAD	0x04000000	make code pages swap on load
MTEPORTHOLE	0x08000000	porthole module
MTEMODPROT	0x10000000	Module has shared memory protected
MTENEWMOD	0x20000000	Newly added module
MTEDLLTERM	0x40000000	Gets instance termination
MTESYMLOADED	0x80000000	Set if debugger symbols loaded

#### **mte\_flags2** flag definitions:

Name	Bit Mask	Description
MTEFORMATMASK	0x0003	Module format mask
MTEFORMATR1	0x0000	Module format reserved
MTEFORMATNE	0x0001	Module format NE
MTEFORMATLX	0x0002	Module format LX
MTEFORMATR2	0x0003	Module format reserved
MTESYSTEMDLL	0x0004	DLL exists in system list

MTELOADORATTACH	0x0008	Module under load or attach - for init
MTECIRCLELREF	0x0010	Module circular reference detection
MTEFREEFIXUPS	0x0020	Free system mte's fixup flag d#98488
MTEPRELOADED	0x0040	MTE Preload completed
MTEGETMTEDONE	0x0080	GetMTE already resolved
MTEPACKSEGDONE	0x0100	Segment packed memory allocated
MTE20LIELIST	0x0200	Name present in version20 lie list
MTESYSPROCESSED	0x0400	System DLL already processed
MTEDLLONEXTLST	0x1000	DLL has term routine on exit list #74177 removed - 75809

## Module Table Entry for OS/2 V2.11

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
mte_flags2	+0	2	W	Module flags 2
mte_handle	+2	2	W	the handle for this mte
mte_swapmte	+4	4	D	link to swappable mte
mte_modname	+8	4	A	resident module name (zero extended)
mte_link	+c	4	D	link to next mte
mte_flags1	+10	4	D	Module flags 1
mte_impmodcnt	+14	4	D	Num of entries in Imp Mod Name Tbl
mte_sfn	+18	2	W	file system number for open file
mte_usecnt	+1a	2	W	.EXE only - use count

## Swappable Module Table Entry for OS/2 Warp V4.0

For **SMTE** formats for other versions of OS/2 see:

[SMTE](#) for OS/2 Warp V3.0

### Pointers

MTE field **mte\_swapmte** points to the associated SMTE.

### Locations

Dynamically allocated from the kernel swappable heap except for the SMTE associated with DOSCALLS.DLL.

**\_DosMosMteSwappable** locates the SMTE in OS2KRNL associated with DOSCALLS.DLL.

**VM Owner**Dynamically allocated SMTEs have owner id **ldrmte (0xffa6)****Format**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
smte_mpages	+0	4	D	Module # pages
smte_startobj	+4	4	D	Object # for instruction pointer
smte_eip	+8	4	D	Extended instruction pointer
smte_stackobj	+c	4	D	Object # for stack pointer
smte_esp	+10	4	D	Extended stack pointer
smte_pageshift	+14	4	D	
smte_fixupsize	+18	4	D	Fixup section size
smte_objtab	+1c	4	D	Object table offset
smte_objcnt	+20	4	D	Number of objects in module
smte_objmap	+24	4	D	Object page map offset
smte_itermap	+28	4	D	Object iterated data map offset
smte_rsrctab	+2c	4	D	Offset of Resource Table
smte_rsrcnt	+30	4	D	Number of resource entries
smte_restab	+34	4	D	Offset of resident name table
smte_enttab	+38	4	D	Offset of Entry Table
smte_fpagetab	+3c	4	D	Offset of Fixup Page Table
smte_frextab	+40	4	D	Offset of Fixup Record Table
smte_impmod	+44	4	D	Offset of Import Module Name Table
smte_impproc	+48	4	D	Offset of Imp Procedure Name Tab
smte_datapage	+4c	4	D	Offset of Enumerated Data Pages
smte_nrestab	+50	4	D	Offset of Non-resident Names Table
smte_cbnrestab	+54	4	D	Size of Non-resident Name Table
smte_autods	+58	4	D	Object # for automatic data object
smte_debuginfo	+5c	4	D	Offset of the debugging info
smte_debuglen	+60	4	D	The len of the debug info in bytes
smte_heapsize	+64	4	D	use for converted 16-bit modules
smte_path	+68	4	D	full pathname
smte_semcount	+6c	2	W	Count of threads waiting on MTE semaphore. 0 => semaphore is free
smte_semowner	+6e	2	W	Slot number of the owner of MTE semahore
smte_pfilecache	+70	4	D	Pointer to file cache for Dos32CacheModule
smte_stacksize	+74	4	D	Thread 1 Stack size from the exe header
smte_alignshift	+78	2	W	use for converted 16-bit modules
smte_NEexpver	+7a	2	W	expver from NE header
smte_pathlen	+7c	2	W	length of full pathname

smte_NEexetype	+7e	2	W	exetype from NE header
smte_csepack	+80	2	W	count of segs to pack
smte_major_os	+82	1	B	major os version to lie about
smte_minor_os	+83	1	B	minor os version to lie about

-----

## Swappable Module Table Entry for OS/2 Warp V3.0

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
mte_flags2	+0	2	W	Module flags 2
mte_handle	+2	2	W	the handle for this mte
mte_swapmte	+4	4	D	link to swappable mte
mte_link	+8	4	D	link to next mte
mte_flags1	+c	4	D	Module flags 1
mte_impmodcnt	+10	4	D	Num of entries in Imp Mod Name Tbl
mte_sfn	+14	2	W	file system number for open file
mte_usecnt	+16	2	W	.EXE only - use count
mte_modname	+18	8	B	resident module name (zero extended)

-----

## Object Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0

### Pointers

SMTE field **smte\_objtab** points to the associated OTE for 32-bit modules.

### Locations

Dynamically allocated from the kernel swappable heap except for the SMTE associated with DOSCALLS.DLL.

**dcm\_ote\_start** locates the OTE in OS2KRNL associated with DOSCALLS.DLL.

### VM Owner

Dynamically allocated OTEs have owner id **ldrmte (0xffa6)** and share the same heap block as their SMTE.

### Format

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ote_size	+0	4	D	Object virtual size
ote_base	+4	4	D	Object base virtual address
ote_flags	+8	4	D	Attribute flags

ote_pagemap	+c	4	D	Object page map index
ote_mapsize	+10	4	D	Num of entries in obj page map
ote_resu	+14	4	S	

#### ote\_flags flag definitions:

Name	Bit Mask	Description
OBJREAD	0x00000001L	Readable Object
OBJWRITE	0x00000002L	Writeable Object
OBJEXEC	0x00000004L	Executable Object
OBJRSRC	0x00000008L	Resource Object
OBJDISCARD	0x00000010L	Object is Discardable
OBJSHARED	0x00000020L	Object is Shared
OBJPRELOAD	0x00000040L	Object has preload pages
OBJINVALID	0x00000080L	Object has invalid pages
OBJZEROFIL	0x00000100L	Object has zero-filled pages
OBJRESIDENT	0x00000200L	Object is resident
OBJALIAS16	0x00001000L	16
OBJBIGDEF	0x00002000L	Big/Default bit setting
OBJCONFORM	0x00004000L	Object is conforming for code
OBJIOPL	0x00008000L	Object I/O privilege level
OBJMADEPRIV	0x40000000L	Object is made private for debug (now obsolete)
OBJALLOC	0x80000000L	Object is allocated used by loader

-----

## Segment Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0

#### Pointers

SMTE field **smte\_objtab** points to the associated STE for 16-bit modules.

#### Locations

Dynamically allocated from the kernel swappable heap.

#### VM Owner

Dynamically allocated STEs have owner id **ldrmte (0xffa6)** and share the same heap block as their SMTE.

#### Format

Field Name	Offset	Length	Type	Description
ste_offset	+0	2	W	file offset to segment data

ste_size	+2	2	W	file data size
ste_flags	+4	2	W	type and attribute flags
ste_minsiz	+6	2	W	minimum allocation size
ste_seghdl	+8	2	W	segment handle
ste_selector	+a	2	W	segment selector
ste_fixups	+c	4	D	fixup record storage

#### ste\_flags flag definitions:

Name	Bit Mask	Description
STE_TYPE_MASK	0x0001	segment type field
STE_CODE	0x0000	code segment type
STE_DATA	0x0001	data segment type
STE_PACKED	0x0002	segment is packed
STE_SEMAPHORE	0x0004	segment semaphore
STE_ITERATED	0x0008	segment data is iterated
STE_WAITING	0x0010	segment is waiting on semaphore
STE_SHARED	0x0020	segment can be shared
STE_PRELOAD	0x0040	segment is preload
STE_ERONLY	0x0080	excute only if code segment read only if data segment
STE_RELOCINFO	0x0100	set if segment has reloc records
STE_CONFORM	0x0200	segment is conforming
STE_RING_2	0x0800	ring 2 selector
STE_RING_3	0x0C00	ring 3 selector
STE_HUGE	0x1000	huge segment
STE_PAGEABLE	0x2000	just a page can be faulted in
STE_PRESENT	0x2000	packed segment already loaded
STE_SELALLOC	0x4000	used to indicate sel allocated
STE_GDTSEG	0x8000	used to indicate GTD sel alloc

## Loader Demand Load Data OS/2 Warp V4.0 and OS/2 Warp V3.0

#### Pointers

PTDA field **ptda\_pldrldHead** points to chain of modules loaded by **DosLoadModule** for a given process.

**\_pldrldHeadKernel** points to the head of kernel reference list of LDRDLs.

**Locations**  
Dynamically allocated from the kernel resident heap.

**VM Owner**  
**ldrld (0xffa4)**

**Format**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
ldrld_pldrldNext	+0	4	D	Pointer to next ldrld
ldrld_hmte	+4	2	W	handle of loaded module
ldrld_cRef	+6	2	W	Number of times loaded

---

## File System Control Block Reference

The following control blocks are described in this section:

- [File System Control Block Entry \(FSC\)](#)
- [System File Table Entry \(SFT\)](#)
- [Master File Table Entry \(MFT\)](#)
- [Record Lock Record \(RLR\)](#)
- [Volume Parameter Block \(VPB\)](#)
- [Drive Parameter Block \(DPB\)](#)
- [Current Directory Structure \(CDS\)](#)
- [File System Buffer \(BUF\)](#)
- [Named Pipe Structures](#)
- [Anonymous Pipe Structures](#)

An overview of the File System Control Blocks follows:

---

## File System Control Block Diagrams

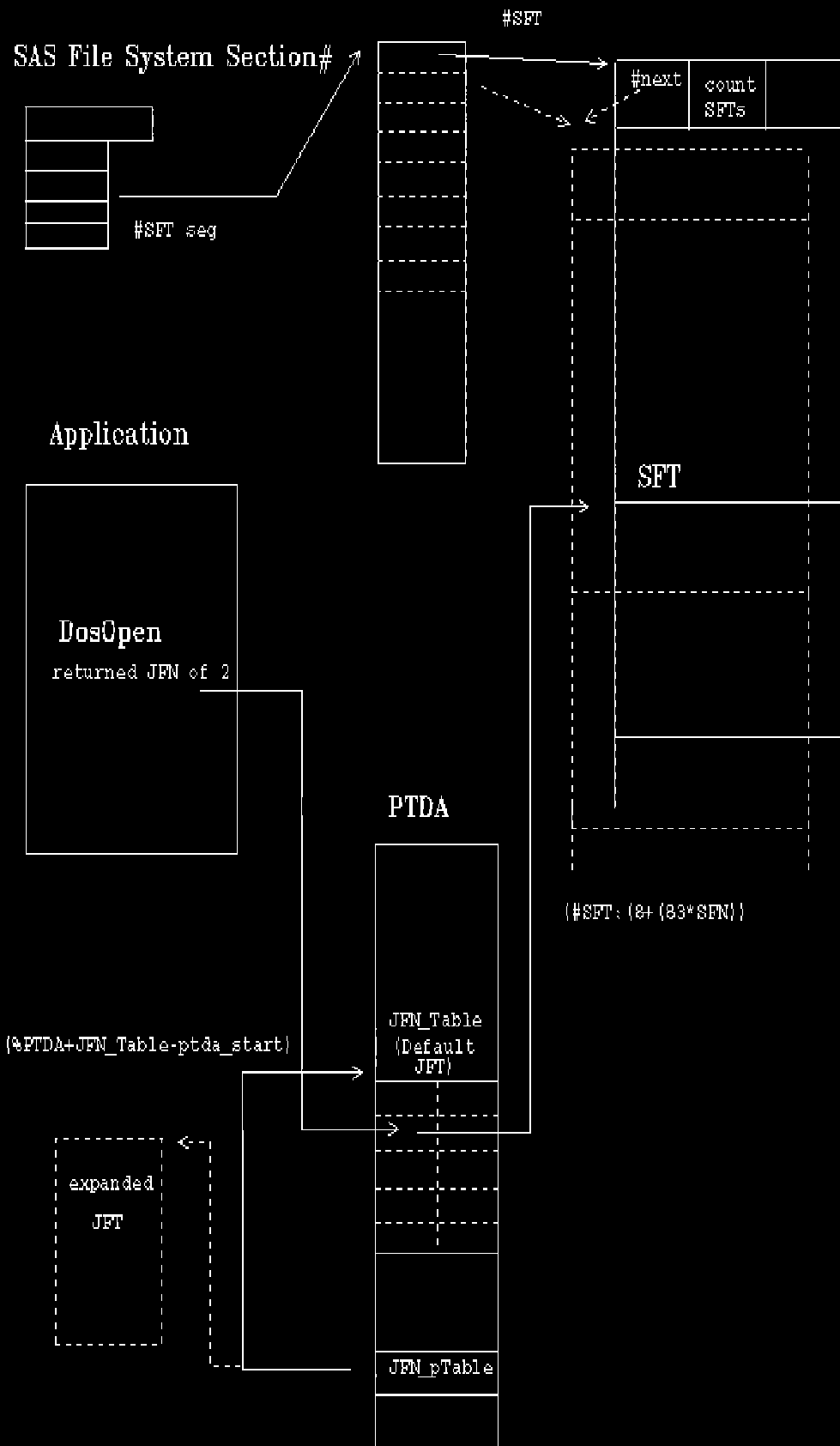
The following diagrams illustrate the relationships between various file system control blocks:

- [Open Files - Application to System](#)
- [Open Files - System View](#)
- [Open Device - System View](#)
- [Shared Files with Locked Ranges](#)
- [Anonymous and Named Pipes](#)

Open Files - Application to System



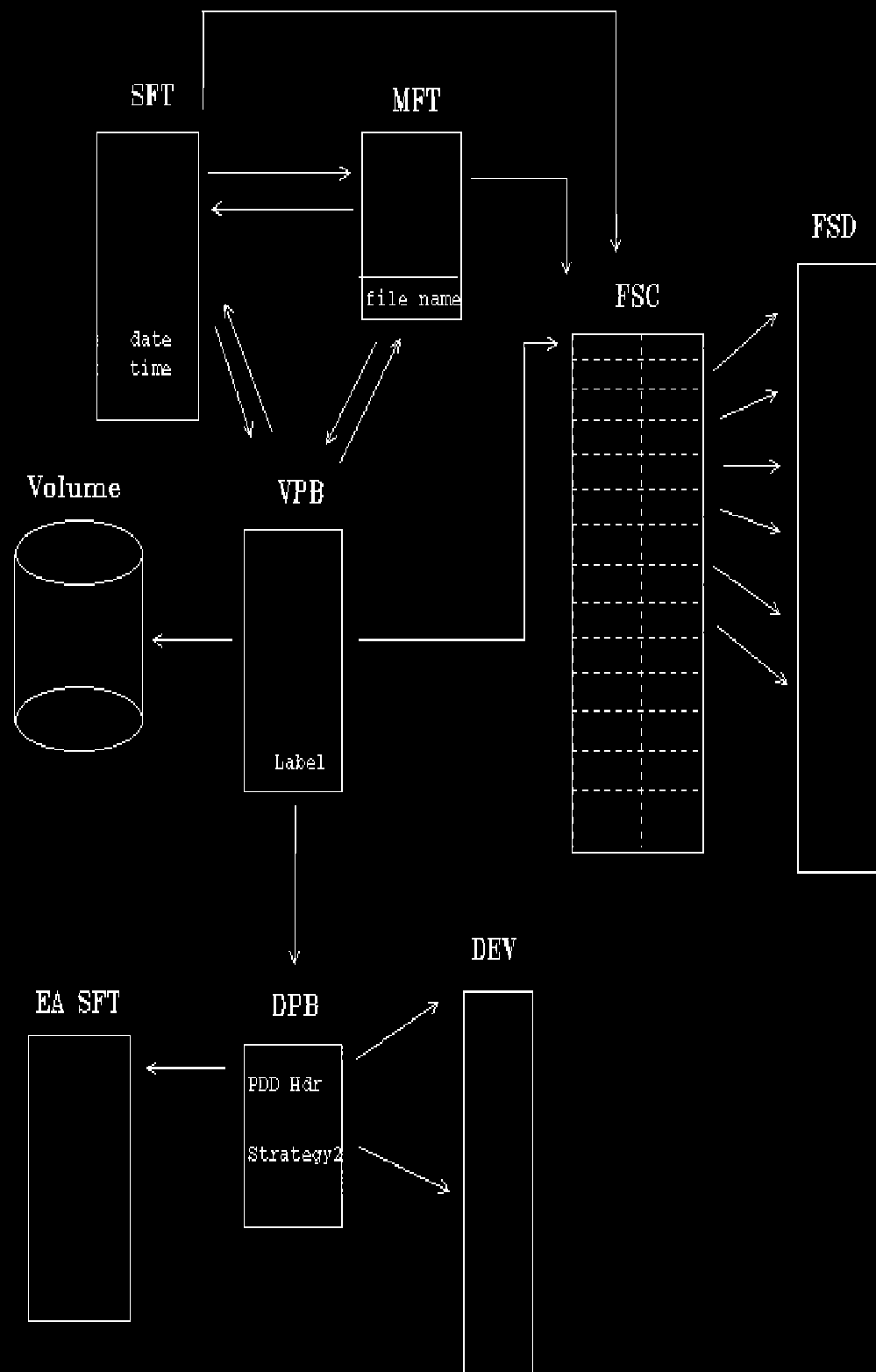
# Open File - Application to System



-----

## Open Files - System View

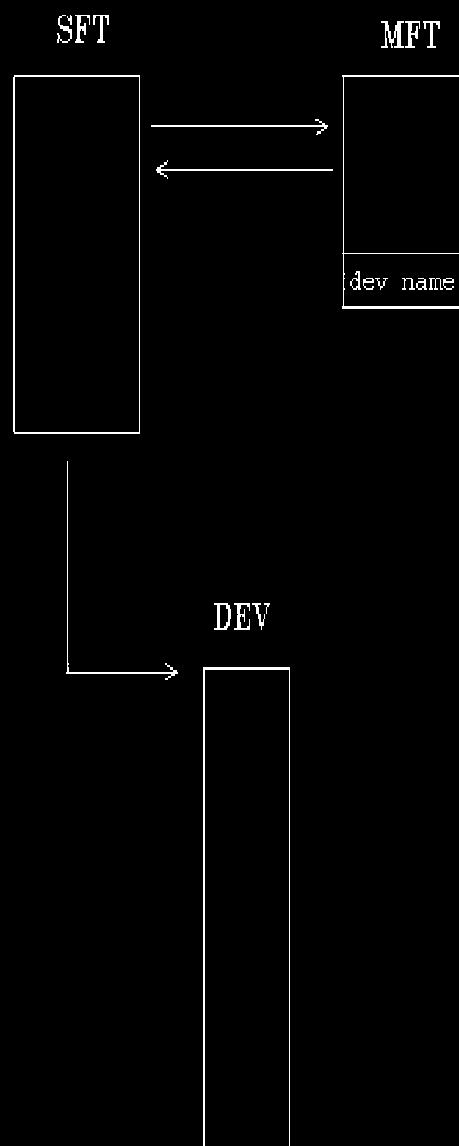
## Open File - System View



-----

Open Device - System View

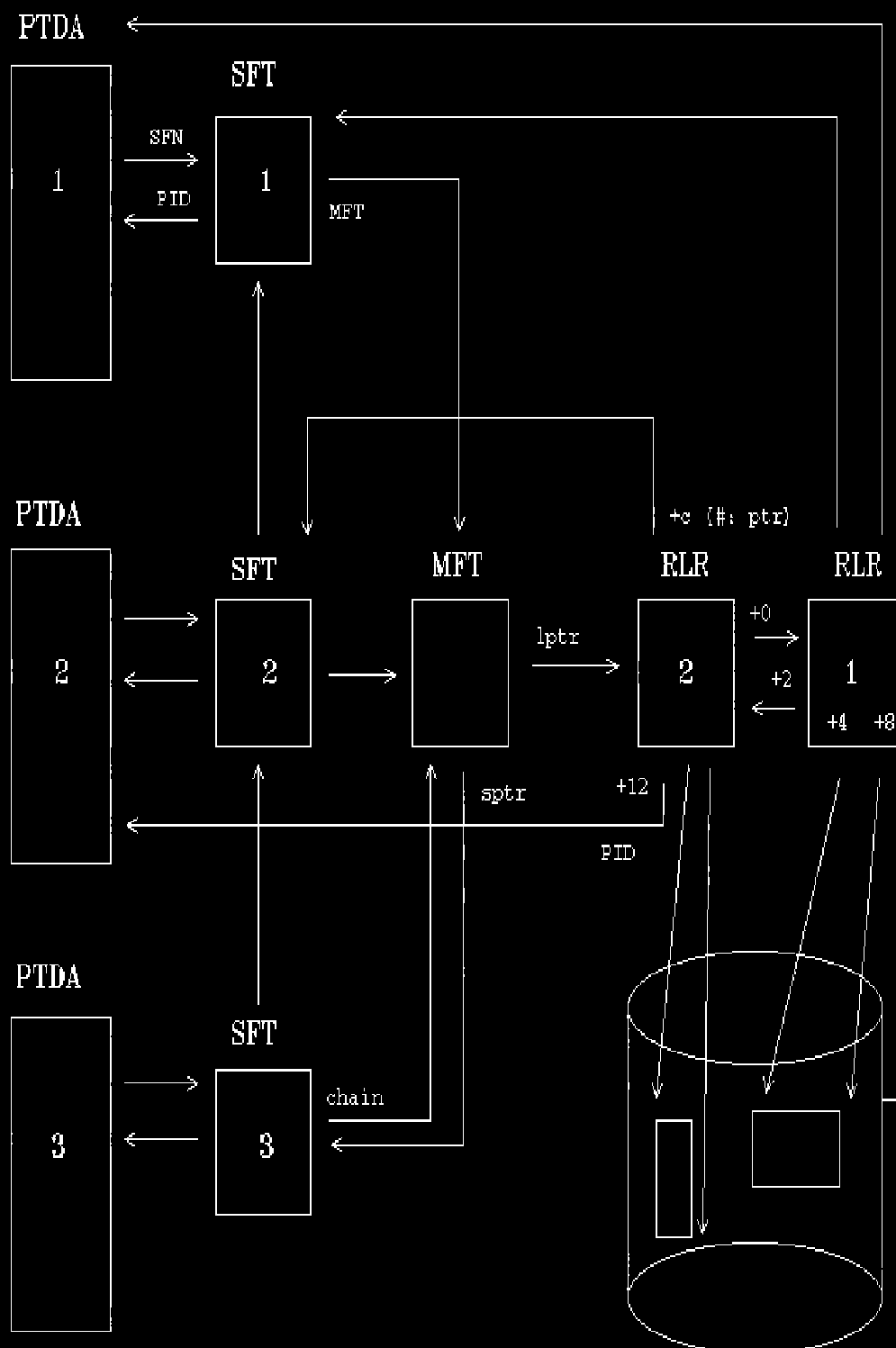
## Open Device - System View



---

## Shared Files with Locked Ranges

# Shared File with 2 Locked Ranges



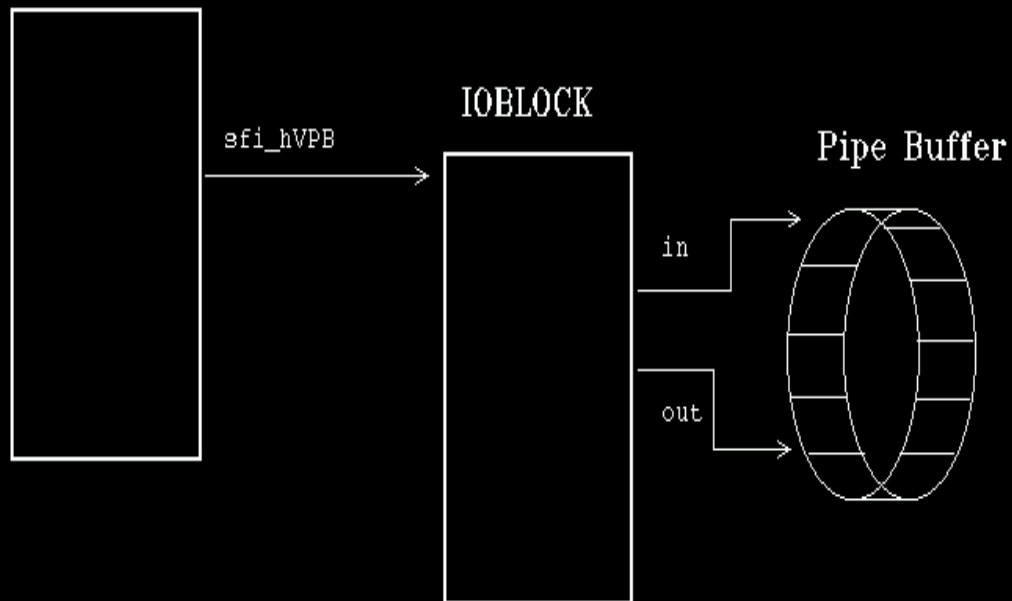
---

## Anonymous and Named Pipes



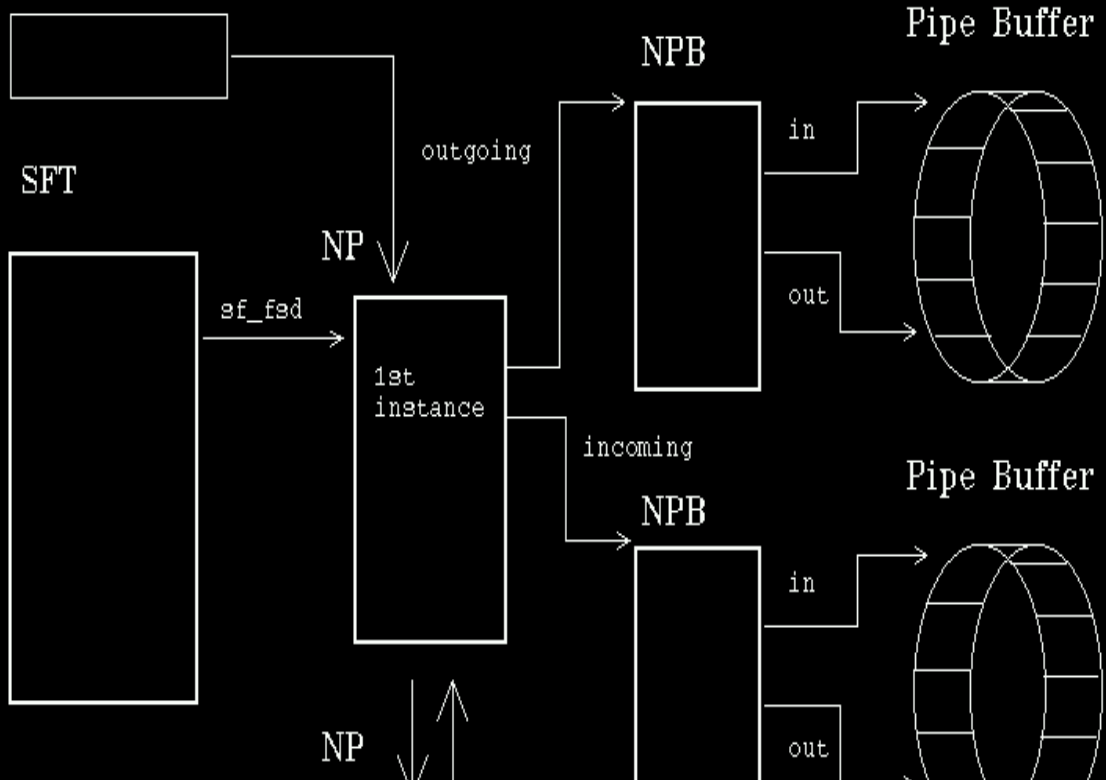
## Anonymous Pipes

SFT



## Named Pipes

NPN



# File System Control Block for OS/2 Warp V4.0 and OS/2 Warp V3.0

## Pointers

SFT field **sf\_FSC** points to the associated FSC\_ENTRY.

VPB field **vpb\_FSC** points to the associated FSC\_ENTRY.

CDS field **cd\_ownerFSC** points to the associated FSC\_ENTRY.

SAS field **SAS\_dd\_FSC** contains the selector for FSCSEG.

**GDT\_FSC** locates the GDT descriptor for the FSCSEG.

## Locations

Dynamically allocated from the kernel resident heap.

## VM Owner

**fsc (0xff95).**

## Format

### FSCSEG

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
fss_Limit	0	2	W	Offset PAST last allocated byte
fss_ShutdownFlags	2	2	W	flags for shutdown
fss_SDWaitCount	4	2	W	number of processes pending before
fss_pad	6	2	W	shutdown can commence (DWORD align)

### FS\_ENTRY

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
FS_ATTRIBUTE	0	4	D	-> FSD attribute. (in FSD memory)
FS_NAME	4	4	D	-> FSD name. (in FSD memory)
FS_ATTACH	8	4	D	DosQFsAttach, DosFsAttach
FS_CHDIR	C	4	D	DosChdir
FS_CHGFILEPTR	10	4	D	DosChgFilePtr
FS_CLOSE	14	4	D	DosClose
FS_COPY	18	4	D	DosCopy
FS_DELETE	1C	4	D	DosDelete
FS_EXIT	20	4	D	DosExit
FS_FILEATTRIBUTE	24	4	D	DosFileInfo, DosSetFileMode
FS_FILEINFO	28	4	D	DosQFileInfo, DosSetFileInfo
FS_FILEIO	2C	4	D	DosFileIO

FS_FINDCLOSE	30	4	D	DosFindClose
FS_FINDFIRST	34	4	D	DosFindFirst
FS_FINDFROMNAME	38	4	D	DosFindFromName-Private to server
FS_FINDNEXT	3C	4	D	DosFindNext
FS_FINDNOTIFYCLOSE	40	4	D	DosFindNotifyClose
FS_FINDNOTIFYFIRST	44	4	D	DosFindNotifyFirst
FS_FINDNOTIFYNEXT	48	4	D	DosFindNotifyNext
FS_FSINFO	4C	4	D	DosQFsInfo, DosSetFsInfo
FS_INIT	50	4	D	-- No corresponding API
FS_IOCTL	54	4	D	DosDevIoctl
FS_MKDIR	58	4	D	DosMkdir
FS_MOUNT	5C	4	D	-- No corresponding API
FS_MOVE	60	4	D	DosMove
FS_NEWSIZE	64	4	D	DosNewsize
FS_NMPIPE	68	4	D	All named pipe related API's
FS_OPENCREATE	6C	4	D	DosOpen
FS_PATHINFO	70	4	D	DosQPathInfo, DosSetPathInfo
FS_PROCESSNAME	74	4	D	-- No corresponding API
FS_READ	78	4	D	DosRead, DosReadAsync
FS_RMDIR	7C	4	D	DosRmdir
FS_SETSWAP	80	4	D	-- No corresponding API
FS_WRITE	84	4	D	DosWrite, DosWriteAsync
FS_OPENPAGEFILE	88	4	D	init time only
FS_ALLOCATEPAGESPACE	8C	4	D	size swap file
FS_CANCELLOCKREQUEST	90	4	D	DosCancelLockRequest
FS_FILELOCKS	94	4	D	DosSetFileLocks
FS_VERIFYUNCNAME	98	4	D	Used to save function addresses
FS_COMMIT	9C	4	D	DosBufReset, DosClose
FS_DOPAGEIO	A0	4	D	perform paging
FS_FSCTL	A4	4	D	DosFsCtl
FS_FLUSHBUF	A8	4	D	DosBufReset
FS_SHUTDOWN	AC	4	D	DosShutdown
FS_SDCHGFILEPTR	B0	4	D	Used to save function addresses
FS_SDFSINFO	B4	4	D	at shutdown time. These functions
FS_SDREAD	B8	4	D	will only be called by shutdown
FS_SDWRITE	BC	4	D	filters.

#### FS\_ATTRIBUTE flag definitions

Name	Bit Mask	Description
FS_ATTR_REMOTE	0x0001	0 = local FSD, 1 = remote FSD
FS_ATTR_UNC	0x0002	0 = normal, 1 = this is UNC FSD
FS_ATTR_LOCKINFO	0x0004	0 = no notice, 1=notify filelocks
FS_ATTR_LVL7	0x0008	0 = no level 7 requests, 1 = yes
FS_ATTR_PIPESVR	0x0010	0 = don't FSD on PIPE req, 1 = yes
FS_ATTR_VERNO	0x7000	bits 28-30 version no
FS_ATTR_EA	0x8000	bit 31 -> 1 = extended attribute

#### **FS\_COMMIT flag definitions**

Name	Bit Mask	Description
FS_COMMIT_ALL	2	all handles commit
FS_COMMIT_ONE	1	one handle commit

#### **Euqates for close type**

Name	Bit Mask	Description
FS_CL_ORDINARY	0	ordinary close
FS_CL_FORPROC	1	final close for process
FS_CL_FORSYS	2	final close for system

#### **fscnameentstruc**

Field Name	Offset	Length	Type	Description
FSCNmEnt_Emulation	0	4	D	
FSCNmEnt_Group	4	1	B	
FSCNmEnt_NameLen	5	1	B	
FSCNmEnt_ProcName	6	1	B	

#### **mFS\_ENTRY**

Field Name	Offset	Length	Type	Description
MFS_CHGFILEPTR	0	4	D	DosChgFilePtr
MFS_CLOSE	4	4	D	DosClose
MFS_INIT	8	4	D	-- No corresponding API
MFS_OPEN	C	4	D	DosOpen
MFS_READ	10	4	D	DosRead, DosReadAsync
MFS_TERM	14	4	D	DosRead, DosReadAsync

#### **uncfscentrstruc**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
UNCfsc_VpbID	0	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	4	4	D	Hold Seg:ofs to UNC FSD's FSC
UNCfsc_hVPB	8	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	A	2	W	Does this entry contain UNC FSD Info?

#### **uncliststruc**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
UNCfsc_1	0	C	B	
UNCfsc_VpbID	0	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	4	4	D	Hold Seg:ofs to UNC FSD's FSC
UNCfsc_hVPB	8	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	A	2	W	Does this entry contain UNC FSD Info?
UNCfsc_2	C	C	B	
UNCfsc_VpbID	C	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	10	4	D	Hold Seg:ofs to UNC FSD's FSC
UNCfsc_hVPB	14	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	16	2	W	Does this entry contain UNC FSD Info?
UNCfsc_3	18	C	B	
UNCfsc_VpbID	18	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	1C	4	D	Hold Seg:ofs to UNC FSD's FSC
UNCfsc_hVPB	20	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	22	2	W	Does this entry contain UNC FSD Info?
UNCfsc_4	24	C	B	
UNCfsc_VpbID	24	4	D	Unique ID UNC VPB
UNCfsc_FSCptr	28	4	D	Hold Seg:ofs to UNC FSD's FSC
UNCfsc_hVPB	2C	2	W	Handle to VPB in VPB Seg(offset)
UNCfsc_Active	2E	2	W	Does this entry contain UNC FSD Info?

-----

## System File Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0

## Pointers

MFT field **mft\_sptr** points to the associated sf\_entry.

RLR field **rlr\_sptr** points to the associated sf\_entry.

SAS field **SAS\_file\_SFT** contains the selector for the SFT segment table.

NP field **np\_ssft** points to the server SFT for a named pipe.

NP field **np\_csft** points to the client SFT for a named pipe.

**GDT\_SFT** locates the GDT descriptor for the SFT segment table.

## Locations

Dynamically allocated from the system arena.

## VM Owner

**sft (0xffa1).**

## Format

### SFT

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
sft_link	0	2	W	selector for next chunk of table
sft_count	2	2	W	number of entries in this block
sft_handle	4	2	W	handle of segment holding this block
sft_inshutdown	6	2	W	flags for shutdown

### sf\_entry

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
sf_ref_count	0	2	W	number of processes sharing entry
sf_usercnt	2	2	W	For files: number of threads waiting for access to sf_entry. For devices: number of threads using this sf_entry.
reserved	4	1	B	Used to be attr of file - moved to * independent part of the SFT for general * access
sf_flags	5	2	W	Bits 8-15 Bit 15 = 1 if remote file = 0 if local file or device Bit 14 = not used Bit 13 = Pipe bit Bit 12 = FCB bit = 1 if fcb sft = 0 if normal sft Bit 11 = if Pipe, = 0 if anonymous pipe = 1 if named pipe Bit 10 == sf_inuse = sf_entry is in use by some thread, ie busy Bit 9 == sf_want = some thrd blocked waiting to use the sf_entry Bit 8 == sf_noJFN, no handle allocated for sft Bits 0-7 (old FCB_devid bits) If remote file or local file, bit 6=0 if dirty Device ID number, bits 0-5 if local file. bit 7=0 for local file =1 for local I/O device If local I/O device, bit 6=0 if EOF (input)

Bit 5=1 if Raw mode  
 Bit 0=1 if console input device  
 Bit 1=1 if console output device  
 Bit 2=1 if null device  
 Bit 3=1 if clock device

sf_flags2	7	2	W	
sf_devptr	9	4	D	Not used if local file, points
sf_FSC	D	4	D	Pointer to the file system control
sf_cookie	11	4	D	server's per-file id (for oplock support)
sf_chain	15	4	D	16:16 Link to next SFT
sf_MFT	19	4	D	32 bit FLAT pointer to MFT entry
sf_fsd	1D	32	S	File system dependent section
sf_fsi	4F	2a	S	File system independent section
sf_plock	79	2	W	16 bit offset to first pending LOCK record
sf_NmPipeSfn	7b	2	W	SFN of named pipe for spooled files
sf_codepage	7d	2	W	current codepage (font) for data in file
sf_LockID	7f	4	D	lock-id for protected file-handle access

#### sf\_flags flag definitions

Name	Bit Mask	Description
SF_ISNET	0x8000	True if SFT is for remote
SF_PIPE	0x2000	Anonymous Pipe
SF_FCB	0x1000	True if SFT is for an FCB
SF_NMPIPE	0x0800	true if name pipe
SF_INUSE	0x0400	True if sf entry is in use by some thread, that is, busy
SF_BLOCKED	0x0200	True if some thread is blocked waiting to use the sf entry
SF_NOJFN	0x0100	True if no handle alloc'ed for SFT

#### sf\_flags2 flag definitions

Name	Bit Mask	Description
SF_FORMAT_MOUNT	0x8000	True if a format mount was done, and still in effect
SF_BEGINFORMAT_FAILED	0x4000	True if a beginformat ioctl failed
SSF2_LDRBINARYSEM	0x2000	'ON' if SFT owned by some thread
SF_SVRDR	0x1000	serving pipe redirection in effect
SFF2_LOCKED_DRIVE	0x0800	A LOCK was issued on this direct access handle to lock the drive.
SFF2_SPOOLED	0x0400	File is spooled
SFF2_DATAWRITTEN	0x0200	Data written to file

SFF2_Consistency	0x0180	consistency bits
SFF2_CANCELJOB	0x0040	spool job has been canceled*/ ;whs
SFF2_NONSPOOLED	0x0020	File is nonspooled; going to printer
SFF2_STPTHINFDN	0x0010	SetPathInfo done, don't set archive
sff2_RA_ON	0x0008	Readahead started
sff2_UNC	0x0004	UNC object
sff2_isfree	0x0002	this SFT is on free list (unused)
sff2_RA_BIG	0x0001	Big Readahead

### sfdFATFS

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
sfdFAT_firFILEclus	0	2	W	First cluster of file (bit 15 = 0)
sfdFAT_cluspos	2	2	W	Position of last cluster accessed
sfdFAT_lstclus	4	2	W	Last cluster accessed
sfdFAT_dirsec	6	4	D	Sector # of directory sector for this file
sfdFAT_dirpos	A	1	B	Offset of this entry in the above
sfdFAT_EAHandle	B	2	W	starting cluster of EAs
sfdFAT_name[11]	D	B	B	
sfdFAT_bRAREads	18	4	D	# of consecutive reads within range
sfdFAT_bRABigReads	1C	4	D	# of consecutive big reads
sfdFAT_fldMask	20	4	D	Unique File Dirty Mask
sfdFAT_pSFT	24	4	D	Linear address of SFT
sfdFAT_ulNextRA	28	4	D	Position where next rahead starts
sfdFAT_bBufRun	2C	4	D	Number of sectors in rahead run
sfdFAT_LastFATSec	30	2	W	last FAT sector added to chain

### sftfsd

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
sfd_work[50]	0	32	B	
			B	

### sftfsi

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
sfi_mode	0	2	W	mode of access or high bit on if FCB
sfi_mode2	2	2	W	additional openmode bits for DosOpen2
sfi_hVPB	4	2	W	handle of volume
sfi_ctime	6	2	W	Creation time of file



sfi_cdate	8	2	W	Creation date of file
sfi_atime	A	2	W	Time of last access of file
sfi_adata	C	2	W	Date of last access of file
sfi_mtime	E	2	W	Time of last modification of file
sfi_mdate	10	2	W	Date of last modification of file
sfi_size	12	4	D	Size associated with file
sfi_position	16	4	D	Read/Write pointer or LRU count for FCBs
sfi_UID	1A	2	W	User ID
sfi_PID	1C	2	W	Process ID
sfi_PDB	1E	2	W	Process Data Block
sfi_selfsfn	20	2	W	SFN of this sf_entry, used to speed
sfi_tstamp	22	1	B	update time stamp flags; see ST_ equs
sfi_type	23	2	W	file/device/named-pipe/FCB
pPVDBFil	25	4	D	performance counter data block pointer
sfi_DOSattr	29	1	B	DOS attributes of file(sys,hid,r/o,arch

**sfi\_type** flag definitions.

Name	Bit Mask	Description
SType_FILE	0x0000	file
SType_DEVICE	0x0001	device
SType_NMPIPE	0x0002	named pipe
SType_FCB	0x0004	SFT is for an FCB

## Master File Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0 ALLSTRICT kernel

For **MTE** formats for other versions of OS/2 see:

**MTE** for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

### Pointers

SFT field **sf\_MFT** points to the associated MFT entry.

SAS field **SAS\_file\_MFT** points to the PTREE head for the MFT PTREE.

### Locations

Dynamically allocated from the kernel resident heap.

### VM Owner

**mft (0xff9e).**

### Format

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
mft_ksem	+0	10	S	multi read/single write semaphore
mft_lptr	+10	2	W	16 bit offset to first LOCK record
mft_sptr	+12	4	D	16 bit FAR pointer to first SFT in chain
mft_pCMap	+16	4	D	32 bit FLAT pointer to cluster map
mft_CMapKSem	+1a	10	S	semaphore for access to pCMap
mft_opflags	+2a	2	W	oplock flags
mft_serl	+2c	2	W	serial number for FCB checking
mft_flags	+2e	2	W	general purpose MFT flags
mft_signature	+30	2	W	for sanity check
mft_hvpb	+32	2	W	handle of vpb
mft_name	+34	1	B	start of name string (zero terminated)

#### **mft\_flags** flag definitions

Name	Bit Mask	Description
mft_pagerheap	0x0001	MFT is allocated on pager heap
MFT_DEFAULTHEAP	0x0	MFT is allocated on kernel (heap default MFT heap)

#### **mft\_opflags** flag definitions

Name	Bit Mask	Description
mft_opnolock	0	no oplock or opbatch on file
mft_oplock	1	oplock on file
mft_opbatch	2	opbatch on file
mft_opbreak	4	oplock/batch cleanup in process
mft_opbreakfailed	8	oplock/batch cleanup failed

-----

## Master File Table Entry for OS/2 Warp V4.0 and OS/2 Warp V3.0 RETAIL kernel

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
mft_ksem	+0	C	S	multi read/single write semaphore
mft_lptr	+c	2	W	16 bit offset to first LOCK record

mft_sptr	+e	4	D	16 bit FAR pointer to first SFT in chain
mft_pCMap	+12	4	D	32 bit FLAT pointer to cluster map
mft_CMapKSem	+16	C	S	semaphore for access to pCMap
mft_opflags	+22	2	W	oplock flags
mft_serl	+24	2	W	serial number for FCB checking
mft_flags	+26	2	W	general purpose MFT flags
mft_hvpb	+28	2	W	handle of vpb
mft_name	+2a	1	B	start of name string (zero terminated)

-----

## Record Lock Record for OS/2 Warp V4.0 and OS/2 Warp V3.0

### Pointers

MFT field **mft\_lptr** contains the offset within the RLR segment to the first RLR associated with the MFT.

**GDT\_RLR** locates the GDT descriptor for the RLR segment.

### Locations

Dynamically allocated from the system arena.

### VM Owner

**fsreclok (0xff47).**

### Format

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
rlr_next	+0	2	W	16 bit offset to next RLR. 0 if end
rlr_prev	+2	2	W	16 bit offset to prev RLR. 0 if SFT
rlr_fba	+4	4	D	offset of first byte of locked region
rlr_lba	+8	4	D	offset of last byte of locked region
rlr_sptr	+c	4	D	16:16 FAR pointer to SFT
rlr_UID	+10	2	W	lock issuer's user ID
rlr_PID	+12	2	W	lock issuer's process ID
rlr_PDB	+14	2	W	lock issuer's PDB, 0 for non-3xBox
rlr_flags	+16	1	B	flags

### rlr\_flags flag definitions

Name	Bit	Mask	Description
RLR_EXCLUSIVE	0		
RLR_SHARED	1		

RLR\_WAITING 2  
RLR\_CANCELLOCK 4

-----

# Volume Parameter Block for OS/2 Warp V4.0 and OS/2 Warp V3.0

## Pointers

SFT field **sfi\_hVPB** contains the offset within the VPB segment of the associated VPB.  
MFT field **mft\_hVPB** contains the offset within the VPB segment of the associated VPB.  
DPB field **dpb\_hVPB** contains the offset within the VPB segment of the associated VPB.  
CDS field **cdi\_hVPB** contains the offset within the VPB segment of the associated VPB.  
**GDT\_VPB** locates the GDT descriptor for the VPB segment.

## Locations

VPB segment is dynamically allocated from the kernel resident heap.

## VM Owner

**vpb (0xffa2).**

## Format

### vpb

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
vpb_flink	0	2	W	handle of forward link
vpb_blink	2	2	W	handle of back link
vpb_ref_count	4	2	W	count of objects that point to VPB
vpb_search_count	6	2	W	count of searches that point to VPB
vpb_first_access	8	1	B	initialized to -1 to force a media
vpb_signature	9	2	W	Signature specifying VPB validity
vpb_flags	B	1	B	flags (bits 7,6,3-0 defined below)
vpb_fMisc	C	1	B	More flags (bit 7 defined below)
vpb_FSC	D	4	D	Pointer to the file system control block (FSC).
vpb_fsd	11	40	S	File system dependent section
vpb_fsi	51	2C	S	File system independent section

## vpb\_signature values

Name	Bit Mask	Description
VPB_VALID	0x444A	

VPB\_INVALID 0x4A47

#### vpb\_ID values

Name	Bit Mask	Description
UNREAD_ID1	0x4A52	Media unreadable
UNREAD_ID2	0x534E	Media unreadable
DAMAGED_ID1	0x0000	Media damaged but recognised by IFS
DAMAGED_ID2	0x0000	Media damaged but recognised by IFS

#### vpb\_falgs flag definitions

Name	Bit Mask	Description
VPBCHECK	0x01	a volume ID check is going on for this VPB
VPBNEWBOOT	0x02	new format disk
VPBMOUNT	0x04	Mount in progress
VPBFORMATMOUNT	0x08	FormatMount done, not cleared
VPBINVALID	0x10	volume formatted - old vpb invalid
VPBINITCACHE	0x20	Initializing Cache Data
VPBSETVID	0x40	vid set is in progress
VPBALLOCATE	0x80	cluster allocation in progress

#### vpb\_fMisc flag definitions

Name	Bit Mask	Description
VPB_FM_WRITEABLE	0x01	Set if we know volume can be written
VPB_FM_UNKNOWN	0x02	Set if no FATs and not claimed by FSD
VPB_REMOTE_DRIVE	0x04	set for attaches of remote drives
VPB_FM_ALLOCSHWAIT	0x08	Set if somebody wants alloc access so that they can get some disk clusters for this volume
VPB_FM_ALLOCEXWAIT	0x10	excl.access wait for somebody who wants to release some clusters
VPB_FM_INITCACHE_ERROR	0x20	Error initializing cache
VPB_FM_INITCACHE_DONE	0x40	

#### vpbfsd

Field Name	Offset	Length	Type	Description
vpd_work[64]	0	40	B	
			B	

## vpbfsl

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
vpi_ID	0	4	D	32 bit unique ID of file (See UNREAD_IDx, DAMAGED_IDx )
vpi_pDPB	4	4	D	Drive volume is in
vpi_cbSector	8	2	W	Size of physical sector in bytes
vpi_totsec	A	4	D	Total number of sectors on medium
vpi_trksec	E	2	W	Sectors per track on medium
vpi_nhead	10	2	W	Number of heads in device
vpi_text[12]	12	C	B	
vpi_pDCS	1E	4	D	device capability struc
vpi_pVCS	22	4	D	volume characteristic struc
vpi_drive	26	1	B	drive (0=A)
vpi_unit	27	1	B	unit
vpi_flags	28	2	W	flags for memory restrictions

## vpdFATFS

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
vpdFAT_cluster_mask	0	1	B	Sectors/cluster - 1
vpdFAT_cluster_shift	1	1	B	Log2 of sectors/cluster
vpdFAT_first_FAT	2	2	W	Starting record of FATs
vpdFAT_FAT_count	4	1	B	Number of FATs for this drive
vpdFAT_root_entries	5	2	W	Number of directory entries
vpdFAT_first_sector	7	4	D	First sector of first cluster
vpdFAT_max_cluster	B	2	W	Number of clusters on drive + 1
vpdFAT_FAT_size	D	2	W	Number of records occupied by FAT
vpdFAT_dir_sector	F	4	D	Starting record of directory
vpdFAT_media	13	1	B	Media byte (duplicate of VPB)
vpdFAT_next_free	14	2	W	Cluster # of last allocated cluster
vpdFAT_free_cnt	16	2	W	Count of free clusters, -1 if unknown
vpdFAT_FATentrysize	18	1	B	12 or 16 - can you guess why ??? @@
vpdFAT_IDsector	19	4	D	sector number of ID
vpdFAT_minEOF	1D	2	W	minimum EOF cluster value: 12-bit -> FF8, 16-bit -> FFF8
vpdFAT_access	1F	2	W	whether rmdir XOR mov dir XOR (chdir mkdir OR mov file OR create)* has access to volume
vpdFAT_accwait	21	2	W	who's waiting for access
vpdFAT_alloc	23	2	W	whether disk cluster alloc OR release

vpdFAT_eaflags	25	2	W	flags for EA usage
vpdFAT_eareaders	27	2	W	number of threads with pending reads
vpdFAT_eawaiters	29	2	W	number of threads waiting to run
vpdFAT_eahandles	2B	2	W	number of handles in EAOffTable
vpdFAT_pEASFT	2D	4	D	SFT for "EA DATA. SF"
vpdFAT_pBadSector	31	4	D	Ptr for Bad sectors data
vpdFAT_pClusBitMap	35	4	D	Ptr to free cluster bit map
vpdFAT_pNextFreeBitMap	39	4	D	Ptr to next free bit map position
vpdFAT_cNextFreeBitMap	3D	2	W	Count of dwords remaining in bit map

#### vpdFAT\_eaflags flag definitions

Name	Bit Mask	Description
eavpb_fileopen	0x0001	the EA file on this volume is open
eavpb_changing	0x0002	the EA file is changing
eavpb_dooropen	0x0004	the drive door has been opened

-----

## Drive Parameter Block for OS/2 Warp V4.0 and OS/2 Warp V3.0

#### Pointers

VPB field **vpi\_pDPB** points to the associated DPB.

#### Locations

DPB segment is dynamically allocated from the kernel resident heap.

#### VM Owner

**dpb (0xff96).**

#### Format

#### DPB

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
dpb_drive	+0	1	B	Logical drive # assoc with DPB (A=0,B=1,...)
dpb_unit	+1	1	B	Driver unit number of DPB
dpb_driver_addr	+2	4	D	Pointer to driver
dpb_next_dpb	+6	4	D	Pointer to next Drive parameter block
dpb_cbSector	+a	2	W	sector size (for volume checking)
dpb_first_FAT	+c	2	W	sector of 1st FAT (for ancient dev drivers)
dpb_toggle_time	+e	4	D	time of last drive toggle

dpb_hVPB	+12	2	W	handle of volume currently in drive
dpb_media	+14	1	B	most recent media that was in drive
dpb_flags	+15	1	B	synchronization flags (see below)
dpb_drive_lock	+16	2	W	Contains pid if drive locked by process
dpb_strategy2	+18	4	D	strategy2 addr (or 00000000)

#### **dpb\_faigs** flag definitions:

Name	Bit Mask	Description
DPBCHECK	0x10	disk in drive is being removed/checked for VPB
DPBNONREMOV	0x20	1 => drive supports non-removable media
DPBVCRAMDISK	0x40	Ram Disk Driver

#### **DPB3X**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
dpb3x_drive	+0	1	B	Logical drive # assoc with DPB (A=0,B=1,...)
dpb3x_UNIT	+1	1	B	Driver unit number of DPB
dpb3x_sector_size	+2	2	W	Size of physical sector in bytes
dpb3x_cluster_mask	+4	1	B	Sectors/cluster - 1
dpb3x_cluster_shift	+5	1	B	Log2 of sectors/cluster
dpb3x_first_FAT	+6	2	W	Starting record of FATs
dpb3x_FAT_count	+8	1	B	Number of FATs for this drive
dpb3x_root_entries	+9	2	W	Number of directory entries
dpb3x_first_sector	+b	2	W	First sector of first cluster
dpb3x_max_cluster	+d	2	W	Number of clusters on drive + 1
dpb3x_FAT_size	+f	1	B	Number of records occupied by FAT
dpb3x_dir_sector	+10	2	W	Starting record of directory
dpb3x_driver_addr	+12	4	D	Pointer to driver
dpb3x_media	+16	1	B	Media byte
dpb3x_first_access	+17	1	B	This is initialized to -1 to force a media check the first time this DPB is used
dpb3x_next_dpb	+18	4	D	Pointer to next Drive parameter block
dpb3x_next_free	+1c	2	W	Cluster # of last allocated cluster
dpb3x_free_cnt	+1e	2	W	Count of free clusters, -1 if unknown

#### **DPB4X**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	---------------	---------------	-------------	--------------------



dpb4x_drive	+0	1	B	Logical drive # assoc with DPB (A=0,B=1,...)
dpb4x_UNIT	+1	1	B	Driver unit number of DPB
dpb4x_sector_size	+2	2	W	Size of physical sector in bytes
dpb4x_cluster_mask	+4	1	B	Sectors/cluster - 1
dpb4x_cluster_shift	+5	1	B	Log2 of sectors/cluster
dpb4x_first_FAT	+6	2	W	Starting record of FATs
dpb4x_FAT_count	+8	1	B	Number of FATs for this drive
dpb4x_root_entries	+9	2	W	Number of directory entries
dpb4x_first_sector	+b	2	W	First sector of first cluster
dpb4x_max_cluster	+d	2	W	Number of clusters on drive + 1
dpb4x_FAT_size	+f	2	W	Number of records occupied by FAT
dpb4x_dir_sector	+11	2	W	Starting record of directory
dpb4x_driver_addr	+13	4	D	Pointer to driver
dpb4x_media	+17	1	B	Media byte
dpb4x_first_access	+18	1	B	This is initialized to -1 to force a media check the first time this DPB is used
dpb4x_next_dpb	+19	2	D	Pointer to next Drive parameter block
dpb4x_next_free	+1d	2	W	Cluster # of last allocated cluster
dpb4x_free_cnt	+1f	2	W	Count of free clusters, -1 if unknown

-----

## Current Directory Structure for OS/2 Warp V4.0 and OS/2 Warp V3.0

### Pointers

SAS field **SAS\_file\_CDS** contains the selector for CDS RMP segment.

**CDSAddr** locates the RMP handle which contains the selector for the CDS RMP segment.

### Locations

CDS segment is dynamically allocated from the kernel resident heap.

### VM Owner

**cdsrmp (0xff61).**

### Format

#### cddFATFS

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
cddFAT_id	0	2	W	cluster of current dir

### **cdfs**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	---------------	---------------	-------------	--------------------

cdd_work[8]	0	8	S	
-------------	---	---	---	--

### **cdfs**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	---------------	---------------	-------------	--------------------

cdi_hVPB	0	2	W	hVPB for the drive mapped to this CDS
cdi_end	2	2	W	End of assignment
cdi_flags	4	1	B	fs independent flags (see below)
cdi_text[260]	5	104	A	

### **curdir**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
-------------------	---------------	---------------	-------------	--------------------

cd_handle	0	2	W	lookup key for this CDS
cd_pid	2	2	W	PID part of lookup key for handles 1-26
cd_refcnt	4	2	W	reference count CDS's
cd_flags	6	1	B	See below for definitions
cd_devptr	7	4	D	local pointer to DPB or net device
cd_OwnerFSC	B	2	W	Owner FSC.Offst
cd_fsd	D	8	S	File system dependent section
cd_fsi	15	10A	S	File system independent section
cdi_hVPB	15	2	W	hVPB for the drive mapped to this CDS
cdi_end	17	2	W	End of assignment
cdi_flags	19	1	B	fs independent flags (see below)
cdi_text[260]	1A	104	B	
	11E	1	B	

### **cd\_flags** flag definitions

Name	Bit Mask	Description
CD_ISNET	0x80	This CDS is for a remote drive
CD_INUSE	0x40	This CDS is in use
CD_SPLICE	0x20	This CDS is for a JOINed drive
CD_JOIN	CD_SPLICE	This CDS is for a JOINed drive
CD_LOCAL	0x10	This CDS is for a SUBSTed drive
CD_ISPSEUDOCHAR	0x08	This CDS for a pseudo-char dev
CD_ISUNC	0x04	This CDS for a UNC name

## cdi\_flags flag definitions

Name	Bit Mask	Description
CDI_ISVALID	0x80	This CDS contains a valid cd_fsd
CDI_ISROOT	0x40	This CDS is for a root (no cdfsd)
CDI_MEDIASWAPPED	0x20	This CDS may not be valid (forces

-----

# File System Buffer for OS/2 Warp V4.0 and OS/2 Warp V3.0

## Pointers

SAS field **SAS\_file\_Buffers** contains the selector for the file system buffer segment.

**GDT\_Buffers** locates the GDT descriptor for the BUFSEG segment.

## Locations

BUFSEG segment is dynamically allocated from the kernel resident heap.

## VM Owner

**fsbuf (0xff93).**

## Format

### BUFSEG

Field Name	Offset	Length	Type	Description
bs_MRUHead	+0	2	W	Head of MRU buffer list (LRU tail)
bs_MRUTail	+2	2	W	Tail of MRU buffer list (LRU head)
bs_FreeHead	+4	2	W	Head of Free buffer list
bs_Handle	+6	2	W	Handle for virtual memory manager
bs_nBuffers	+8	2	W	Number of buffers in segment
bs_buffsize	+a	2	W	Size of buffer+header, in bytes.
bs_seglimit	+c	2	W	Limit for entire buffer segment.
bs_pStats	+e	2	W	Offset of statistics block (for PROFILE)
bs_offRemMed	+10	2	W	Minimum "legal" offset of buffer for removable media
bs_MaxSec	+12	2	W	Maximum sector size for block device drivers
bs_BigBufBase	+14	2	W	Base of big buffers pool
bs_BigBufMap	+16	2	W	Big buffers usage bit map (bit0 - Buf0)
bs_physBufSeg	+18	4	D	Buffer segment Physical Address

### BUFFINFO

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
buf_next	+0	2	W	Pointer to next buffer in list (-1 = end)
buf_prev	+2	2	W	Pointer to previous buffer in list (-1 = end)
buf_freeLink	+4	2	W	Pointer to next free buffer (-1 = end)
buf_hVPB	+6	1	W	serial number of volume
buf_sector	+8	4	D	Sector number of buffer
buf_wrtcnt	+c	1	B	For FAT sectors, # times sector written out
buf_wrtcntinc	+d	2	W	For FAT sectors, # sectors between each write
buf_flags	+f	1	B	Flags
buf_tid	+10	2	W	thread ID of buffer owner
buf_refcnt	+12	2	W	number of threads using buffer for read
buf_fill	+14	2	W	random debugging information
buf_pad	+16	2	W	Force dword alignment.

#### buf\_faigs flag definitions:

Name	Bit Mask	Description
BUF_DIRTY	0x80	Bit 7 = 1 if buffer dirty
BUF_VISIT	0x40	Bit 6 = 1 if buffer seen in search
BUF_WANT	0x20	Bit 5 = 1 if process waiting for buffer
BUF_BUSY	0x10	Bit 4 = 1 if in use by process
BUF_ISDATA	0x08	Bit 3 = 1 if buffer is DATA
BUF_ISDIR	0x04	Bit 2 = 1 if buffer is DIR
BUF_ISFAT	0x02	Bit 1 = 1 if buffer is FAT
BUF_ATTEMPTING_READ	0x01	Bit 0 = 1 if buffer is in swapper pool

## Named Pipe Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0

### Pointers

SFT field **sf\_fsd** points to the associated NP structure.

**NmpRmpHand** locates the RMP handle that contains the selector for the NPN RMP segment.

NPN field **npn\_link** points to the double linked list of NP structures that are instances of the named pipe.

Instances of named pipes are double-lined by **np\_flink** and **np\_blink**.

NP fields **np\_selector1** and **np\_selector2** point to associated NPB structures.

#### Locations

The NPN RMP is allocated from the kernel swappable heap.

The NP is allocated from the system arena.

The NPB is allocated from the kernel resident heap.

#### VM Owner

NP owner id is **npipenp (0xff31)**.

NPN owner id is **npipenpn (0xff30)**.

NPB owner id is **npipenbuf (0xff9f)**.

#### Format

There are four important data structures associated with named pipes: the SFT corresponding to an open named pipe, a pair of kernel internal data structures describing the pipe and one or two allocated memory segments which contain the data buffers for the pipe.

The parts of the SFT specific to named pipes are:

<code>sf_flags</code>	SF_NMPIPE and SF_PIPE set
<code>sf_np</code>	pointer to pipe info.
<code>sf_pipmod</code>	mode of pipe, per-sft internal state bits

Where:

`sf_np` is defined to be `sf_fsd+0`, the pointer to np structure  
`sf_pipmod` is defined to be `sf_fsd+4`, the mode of pipe, plus internal state

#### NP Named Pipe data structure

The internal data structure for an instance of a pipe. One of these structures is allocated for each open instance of a particular named pipe.

Allocated NP structures are placed on two lists. The first is headed by `ActiveNPList`, with list pointer `np_next` linking together all currently active NP structures.

The second list is headed by the NPN structure defined below and is doubly-linked by the `np_flink` and `np_blink` pointers. This list is used to iterate over all instances of a particular pipe name.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
<code>np_state</code>	+0	1	B	state of pipe
<code>np_refcnt</code>	+1	1	B	SFT reference count for pipe (1 or 2)
<code>np_next</code>	+2	2	W	pointer to next in active list
<code>np_flink</code>	+4	2	W	pointer to next instance of pipe
<code>np_blink</code>	+6	2	W	pointer to previous instance of pipe
<code>np_namkey</code>	+8	2	W	RMP key value for npn structure
<code>np_scnt</code>	+a	1	B	count of servers (max. 1)
<code>np_ccnt</code>	+b	1	B	count of clients (max. 1)
<code>np_selector1</code>	+c	2	W	selector for outgoing data buffer
<code>np_selector2</code>	+e	2	W	selector for incoming data buffer
<code>np_pipmod</code>	+10	2	W	pipe mode specified at creation time
<code>np_flags</code>	+12	2	W	pipe flags
<code>np_ssft</code>	+14	4	D	back pointer to server SFT

np_csft	+18	4	D	back pointer to client SFT
np_timeo	+1c	4	D	default timeout for DosWaitNmPipe
np_ssem	+20	4	D	server end system semaphore
np_ssemkey	+24	2	W	server's semaphore key
np_csem	+26	4	D	client end system semaphore
np_csemkey	+2a	2	W	client's semaphore key

#### **NPN** Named Pipe Name data structure

The following structure contains the common name for the multiple instances of a pipe. Its key value is used as the ProcBlock key for waiters on the pipe. The key value is also used as an RMP key to look up the name record from the NP structure.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
npn_link	+0	4	D	pointer to first instance
npn_key	+4	2	W	unique serial number of name
npn_len	+6	2	W	total length of structure
npn_name	+8	254	A	name of pipe, null terminated

#### **NPB** Named Pipe Buffer data structure

The following variables are used to control the access to a pipe buffer and are part of the allocated buffer for the pipe. In the case of a duplex pipe, two independent data buffers are allocated. Only one buffer will be allocated for a simplex pipe.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
npb_selector	+0	2	W	selector of buffer
npb_first	+2	2	W	base of buffer
npb_in	+4	2	W	next free byte in buffer
npb_out	+6	2	W	next byte of data in buffer
npb_last	+8	2	W	end+1 of buffer
npb_rdlck	+a	2	W	read lock sem.
npb_wtlck	+c	2	W	write lock sem.
npb_rdsem	+e	2	W	read sync sem.
npb_wtsem	+10	2	W	write sync sem.
npb_rdcnt	+12	1	B	count of readers of buffer
npb_wtcnt	+13	1	B	count of writers to buffer
npb_data	+14	2	W	size of data left in pipe

#### **np\_state** allowable values for named pipe state

Internally, byte stream mode pipes store just a collection of bytes in the data buffer. Message stream mode pipes have individual messages preceeded by a word which indicates the size of the message.

Named pipes may be in one of several states depending on the actions that have been taken on it by the server end and client end. The following state/action table summarizes the valid state transitions:

Current state	Action	Next state
---------------	--------	------------

<none>	server MakeNmPipe	DISCONNECTED
DISCONNECTED	server connect	LISTENING
LISTENING	client open	CONNECTED
CONNECTED	server disconn	DISCONNECTED
CONNECTED	client close	CLOSING
CLOSING	server disconn	DISCONNECTED
CONNECTED	server close	CLOSING
<any other>	server close	<pipe deallocated>

A special internal state, LISTEN2 is used when a client open is in progress (since some operations may block). This is treated the same as the LISTENING state except that a new open or wait will not recognize it as an available pipe.

If a server disconnects his end of the pipe, the client end will enter a special state in which any future operations (except close) on the file descriptor associated with the pipe will return an error.

Name	Bit Mask	Description
NP_DISCONNECTED	1	after pipe creation or Disconnect
NP_LISTENING	2	after DosNmPipeConnect
NP_CONNECTED	3	after Client open
NP_CLOSING	4	after Client close
NP_LISTEN2	0x12	internal; client open in progress

**np\_pipmod, sf\_pipmod** bit mask values:

Name	Bit Mask	Description
NP_NBLK	0x8000	non-blocking read/write
NP_NBLKR	0x8000	non-blocking read
NP_NBLKW	0x8000	non-blocking write
NP_SERVER	0x4000	set if server end
NP_WMESG	0x0400	write messages
NP_RMESG	0x0100	read as messages
NP_TIMEOUT	0x3800	Timeout np_sem_blk & np_sem_wait

## Anonymous Pipe Structures for OS/2 Warp V4.0 and OS/2 Warp V3.0

### Pointers

SFT field **sfi\_hvpb** contains the selector that maps IOBLOCK structure.

### Locations

The pipe IOBLOCK is allocated from the kernel heaps.

### VM Owner

**pipe (0xffa0).**

### Format

## **IOBLOCK** Anonymous Pipe data structure

A 'pipe' is a connection between (among) file handles (JFN's). Data written to the 'write end' of the pipe are made available for reading on the 'read end'. The \$Pipe system call creates a pipe and returns two file handles, one for the read end and one for the write end. These handles are manipulated in the same way as normal file handles; they may be 'dup'ed and are inherited in the same way. Data are written into a pipe via a 'write' system call on the write end of the pipe. Likewise, data are read from the pipe via a 'read' call on the read end.

Data that are written to a pipe are captured in a circular buffer. The size of the buffer is specified when the pipe is created; if no size is specified, a default size is used.

The circular buffer is described by an 'ioblock'. The ioblock is the buffer's header; the circular buffer proper follows the ioblock in a heap memory object (mapped by a GDT selector) allocated when the pipe is created. The ioblock contains all of the per-pipe information, such as reader, writer, and reference counts, and also holds the pointers into the circular buffer proper.

The selector that points to the circular buffer is stored in the SFT, at sfi\_hVPB.

When the in and out pointers are equal, the circular buffer is empty. When the in pointer trails the out pointer by 1, the buffer is full. Thus, a 512 byte buffer can hold only 511 bytes; one byte is lost so that full and empty conditions can be distinguished. So that the user can put 512 bytes in a pipe that they created with a size of 512, we allow for this byte lost when allocating the segment.

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
io_inprogcnt	+0	1	B	count of read/wrts in progress
io_refcnt	+1	1	B	count of references
io_rdrCNT	+2	1	B	count of readers
io_wtrcnt	+3	1	B	count of writers
io_selector	+4	2	W	buffer selector
io_first	+6	2	W	ptr to base of circular buffer
io_in	+8	2	W	ptr to next free byte
io_out	+a	2	W	ptr to next byte of data
io_last	+c	2	W	ptr to end+1 of buffer
io_rdlksem	+e	2	W	read lock semaphore
io_wtlksem	+10	2	W	write lock semaphore
io_rdsem	+12	2	W	read sync semaphore
io_wtsem	+14	2	W	write sync semaphore

---

## I/O System Control Block Reference

The following control blocks are described in this section:

[Physical Device Driver Header \(DEV\)](#)

[PDD IRQ Information Blocks \(DIRQ\)](#)

[Virtual Device Driver Entry Point Structures](#)

[Device Driver Request Packets \(REQ\)](#)

[BIOS Parameter Block \(BPB\)](#)

An overview of the I/O System Control Blocks follows:

---



# I/O System Control Block Diagrams

The following diagrams illustrate the relationships between various I/O system control blocks:

[Physical Device Driver Communication](#)

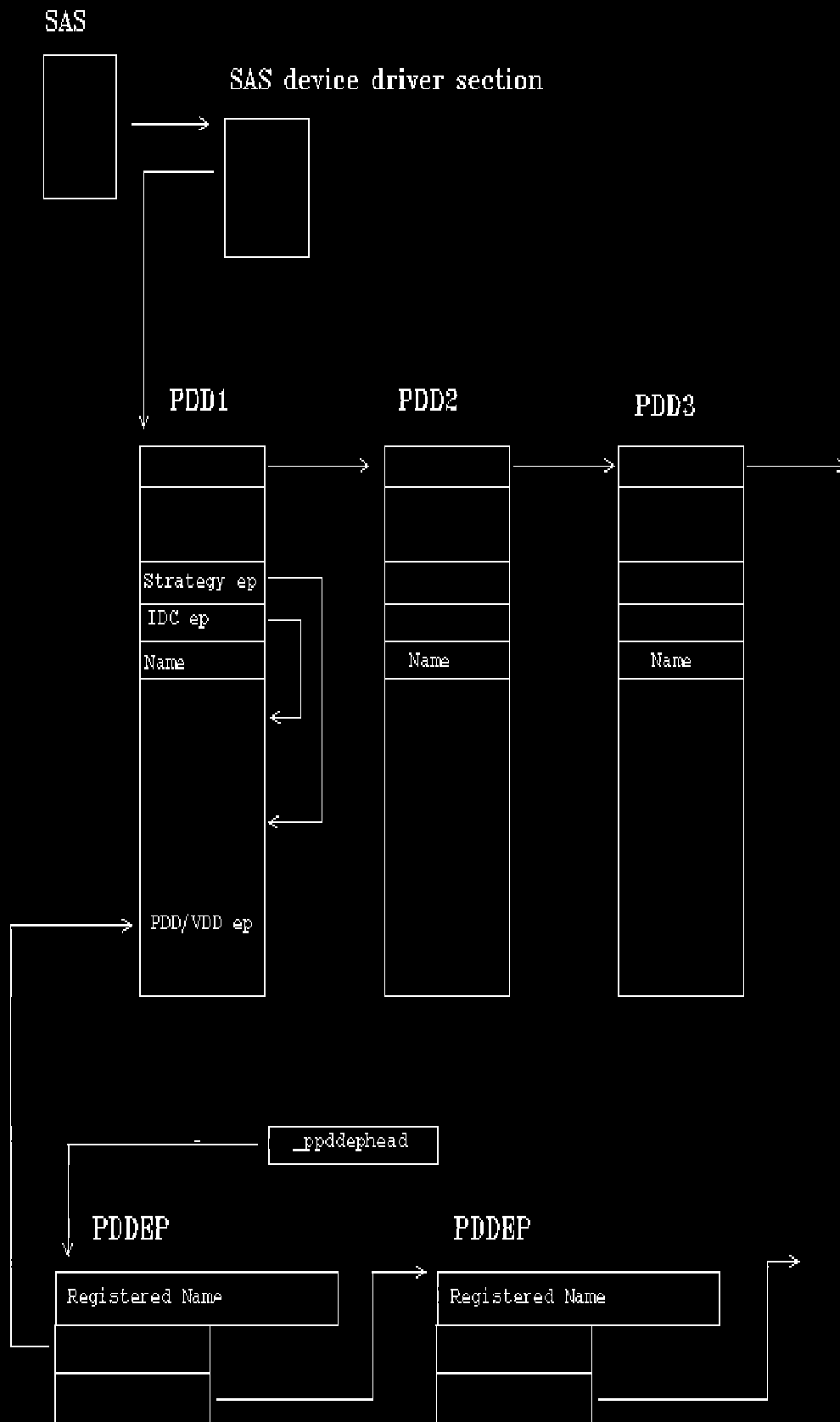
[Physical Device Driver IRQ Sharing](#)

[Virtual Device Driver Communication](#)

-----

## Physical Device Driver Communication

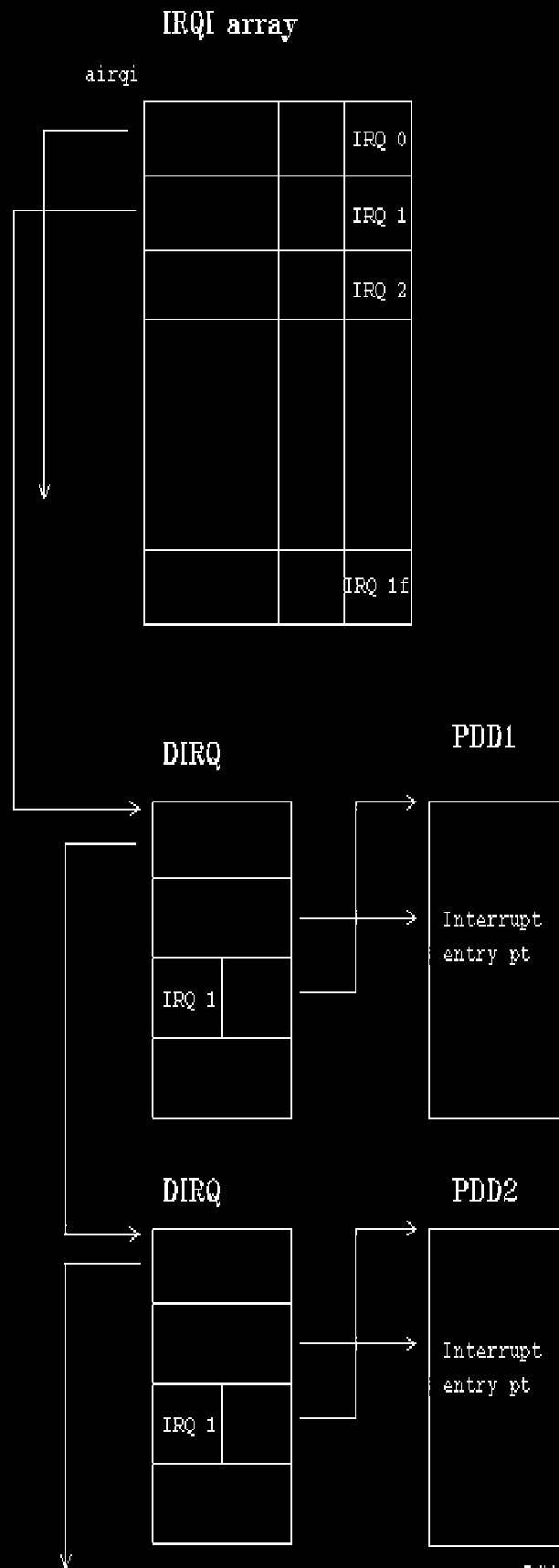
# Physical Device Driver Communication



---

## Physical Device Driver IRQ Sharing

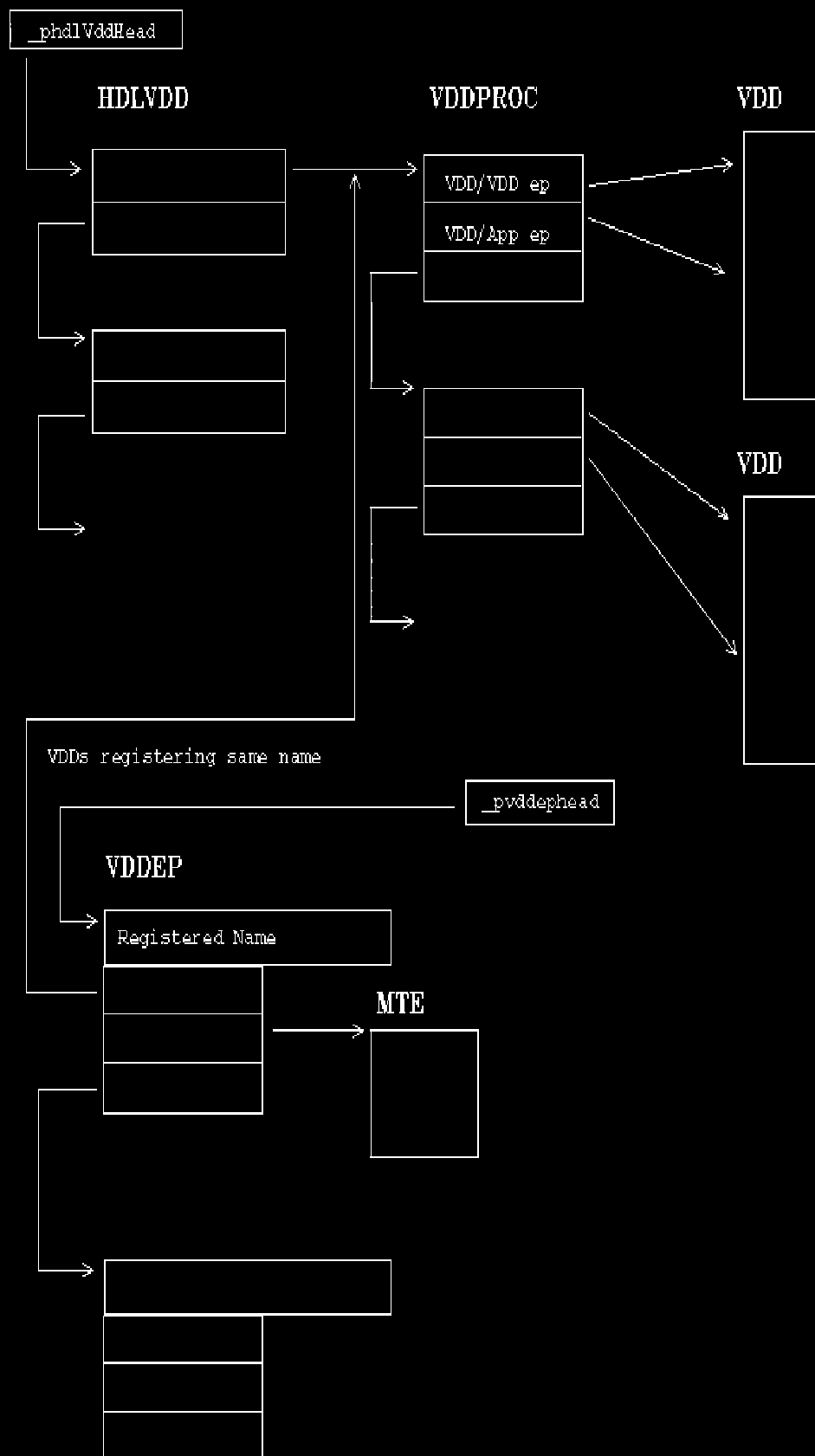
# Physical Device Driver IRQ Sharing



---

## Virtual Device Driver Communication

# Virtual Device Driver Communication



-----

# Physical Device Driver Header (DEV) for OS/2 Warp V4.0 and OS/2 Warp V3.0

**Pointers**

DPB field **dpb\_driver\_addr** points to the associated Physical Device Driver Header.

SFT field **sf\_devptr** points to the associated Physical Device Driver Header.

**Locations**

Built at the beginning of the first module segment of the device driver.

**VM Owner**

**dd1 (0xff50) to dd16 (0xff5f).**

**Format**

Field Name	Offset	Length	Type	Description
SDevNext	+0	4	D	Pointer to next device header
SDevAtt	+4	2	W	Attributes of the device
SDevStrat	+6	2	W	Strategy entry point
SDevInt	+8	2	W	IDC entry point
SDevName	+a	8	A	name (block uses only 1st byte)
SDevProtCS	+12	2	W	Protect-mode CS of strategy entry pt
SDevProtDS	+14	2	W	Protect-mode DS
SDevRealCS	+16	2	W	Real-mode CS of strategy entry pt
SDevRealDS	+18	2	W	Real-mode DS
SDevCaps	+20	4	D	bit map of DD /MM restrictions

**SDevCaps flag definitions**

Name	Bit Mask	Description
DEV_IOCTL2	0x0001	DD can handle dev ioctl2
DEV_16MB	0x0002	DD can handle phys.addresses >16MB
DEV_PARALLEL	0x0004	DD handles parallel port
DEV_ADAPTER_DD	0x0008	DD supports Adapter Dev Driver Intf
DEV_INITCOMPLETE	0x0010	DD can handle CMDInitComplete

**Device Driver Type defininitions**

Name	Bit Mask	Description
DEV_CIN	0x0001	0 2 5 Device is console in

DEV_COUT	0x0002	1 2 5	Device is console out
DEV_NULL	0x0004	2 2 5	Device is the Null device
DEV_CLOCK	0x0008	3 2 5	Device is the clock device
DEV_SPEC	0x0010	4 2	Devices can support INT 29h
DEV_ADD_ON	0x0020	5	Device is add-on driver (BWS)
DEV_IOCTL	0x0040	6 3	Device supports generic ioctl
DEV_FCNLEV	0x0380	9-7 5	Device function level
DEV_30	0x0800	11 2 5	Accepts Open/Close/Removable Media
DEV_SHARE	0x1000	12	Device wants FS sharing checking
DEV_NON_IBM	0x2000	13 2 5	Device is a non IBM device.
DEV_IOCTL	0x4000	14 2	Device accepts IOCTL request
DEV_CHAR_DEV	0x8000	15 2 5	Device is a character device

#### Level definitions for devices

Name	Bit	Mask	Description
DEVLEV_0	0x0000		DOS 3.0 and before
DEVLEV_1	0x0080		DOS 5.0
DEVLEV_2	0x0100		OS/2 v1.2 (new gen ioctl iface)
DEVLEV_3	0x0180		OS/2 v2.0 (support of memory above 16MB)

## PDD IRQ Information Blocks (DIRQ) for OS/2 Warp V4.0 and OS/2 Warp V3.0

#### Pointers

IRQI field **irqi\_pdirqHead** points to the head of a chain of associated DIRQs.

#### Locations

**airqi** locates the table of IRQI entries.

DIRQs are allocated dynamically from the kernel resident heap.

The IRQI array is a static part of the OS2KRNL load module.

#### VM Owners

IRQI owner id: **os2krnl (0xffaa)**.

DIRQ owner id: **intdirq (0xff78)**.

#### Format

#### IRQI

Field Name	Offset	Length	Type	Description
------------	--------	--------	------	-------------



irqi_pdirqHead	+0	4	D	Head of shared DD chain (0 = not set)
irqi_usIRQNum	+4	2	W	IRQ number
irqi_usFlags	+6	2	W	IRQ Flags

#### irqi\_usFlags flag definitions

Name	Bit	Mask	Description
	0x0003		reserved
irqf_fVDM	0x0004		If set, this IRQ is a candidate for routing to a VDM, if it is not claimed by a PDD
irqf_fNPX	0x0008		If set, the IRQ is the NPX interrupt level
irqf_fSharing	0x0010		If set, the IRQ is sharable. If clear the IRQ can not be shared by DD.
irqf_fSys	0x0020		If set, the IRQ is owned by the system and the handler can not be changed or removed by a device driver. Set initially for the slave, IRQ 2.
irqf_fShared	0x0040		If set, the IRQ can be shared by more than 1 DD. This bit reflects the shared parameter of the first dh_SetIRQ issued for this level.

#### DIRQ

Field Name	Offset	Length	Type	Description
dirq_pdirqLink	+0	4	D	Next DIRQ structure in list
dirq_f16pfn	+4	4	D	DD's interrupt handler
dirq_usDS	+8	2	W	DD's data segment
dirq_usIRQNum	+a	2	W	IRQ number
dirq_pdirqFreeList	+c	4	D	list of unset DIRQs

## Virtual Device Driver Entry Point Structures

#### Pointers

**\_pvddepHead** points to the head of a chain of VDDEP structres. One is allocated for each VDD the registers an either or both a VDD/VDD or VDD/OS2 entry point. There entry points are used respectively when either a VDHRequestVDD/VDHOpenVDD or DosRequestVDD/DosOpenVDD call is made.

VDDEP field **vddep\_vddp** points to the associated chain of VDDPROC structures. One is allocated for each VDD the registers entry points under the same name.

**\_phdlVddHead** points to the head of a chain of HDLVDD structures. One is allocated for each open VDD. The handle returned is the address of the associated HDLVDD.

**\_ppddephead** points to the head of a chain of PDDEP structures. One is allocated for each Physical Device Driver that registers an entry point for VDD/PDD communication. The entry point is registered using `DevHlp_RegisterPDD`, and accessed using `VDHRequestPDD`.

#### Locations

VDDEPs, VDDPROCs, HDLVDDs and PDDEPs are allocated dynamically from the kernel resident heap.

#### VM Owner

VDDep owner id: **vddep (0xffd2)**.

VDDPROC owner id: **vddproc (0xffdb)**.

HDLVDD owner id: **vddl (0xffd7)**.

PDDEP owner id: **pddep (0xffda)**.

#### Format

##### VDDEP

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
vddep_szVDD	+0	9	A	VDD Name
vddep_vddp	+9	4	D	VDD entry points (pointer to VDDPROC)
vddep_hmte	+d	4	D	VDD hmte for deregistering if VDD fails
vddep_pvddp	+11	4	D	Next VDD (pointer to next VDDEP)

##### VDDPROC

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
vddproc_pfnvdd	+0	4	D	Entry point for VDD/VDD comm.
vddproc_pfnos2	+4	4	D	Entry point for OS2/VDD comm.
pvddproc	+4	4	D	Entry points registered with same name

##### HDLVDD

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
hdlvdd_pvddproc	+0	4	D	VDD routine to be called (pointer to VDDPROC)
hdlvdd	+4	4	D	Pointer to next VDD handle; NULL if no more

##### **PDDEP**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
pddep_szPDD	+0	9	A	PDD name
pddep_fpfn	+9	4	D	Entry point routine
pddep_ppddep	+d	4	D	Next entry point (PDDEP)

-----

# Device Driver (Strategy 1) Request Packet (REQ) for OS/2 Warp V4.0 and OS/2 Warp V3.0

- Pointers** TCB field **TCBReqPkt** points to the Request Packet pre-allocated to a thread.
- Locations** Allocated from the Request Packet Pool in the System Arena.
- VM Owner** **reqpkt1 (0xff40).**

**Format**

<i>Field Name</i>	<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
Packet	+0	20	S	Device Driver Request Packet
PktLen	+0	1	B	length in bytes of packet
PktUnit	+1	1	B	subunit number of block device
PktCmd	+2	1	B	command code
PktStatus	+3	2	W	status word
PktFlag	+5	1	B	disk driver internal flags
	+6	3	B	reserved
PktDOSLink	+5	4	D	
PktDevLink	+9	4	D	device multiple-request link
PktData	+d	18	S	data pertaining to specific packet
	+d	10	S	Generic IOCTL
GIOCategory	+d	1	B	Category Code
GIOFunction	+e	1	B	Function code
GIOParaPack	+f	4	D	pointer to parameter packet
GIODataPack	+13	4	D	pointer to data packet
GIOSFN	+17	2	W	(used by Spooler?)
GIOParaLen	+19	2	w	length of parameter packet
GIODataLen	+1b	2	W	length of data packet
	+d	c	S	INIT Command for Base DDs (0 and 27)
InitcUnit	+d	1	B	number of units returned
InitpEnd	+e	4	D	pointer to free mem after dev
InitDevHlp	+e	4	D	address of Device Helper router
InitEcode	+e	2	W	size of code segment
InitEdata	+10	2	W	size of data segment
InitParms	+12	4	D	pointer parameters

InitpBPB	+12	4	D	pointer to BPBs
Initdrv	+16	1	B	drive no. assigned to unit 0
	+17	1	B	reserved
InitSysiData (for resident drivers only)	+18	1	B	SysInit's DOSALIAS selector
	+d	d	S	query for extended capability command (0x1d)
	+d	3	B	reserved
DCS_Addr	+10	4	W	16
VCS_Addr	+14	4	W	16
	+d	6	B	Media Check command 1
MedChkmedia	+d	1	B	last media byte seen
MedChkflaga	+e	1	B	-1=change 0=dont know 1=no change
MedChkpVIDa	+f	4	D	pointer to VID
	+d	9	S	build BPB command 2
BldBPBmedia	+d	1	B	media byte
BldBPBbuffer	+e	4	D	scratch buffer
	+d	f	S	Read/Write IO commands 3, 4, 8, 9, 12, 24, 25, 26
IOmedia	+d	1	B	media byte
IOPData	+e	4	D	transfer address
IOcount	+12	2	W	count of bytes/sectors
IOstart	+14	2	W	starting sector (block)
IOPhysRBA	+14	4	D	physical starting sector
IOSFNsRBA	+18	2	W	for device only
PktAdvise	+1a	2	W	for >= v12 only
	+d	4	S	Device Open/Close commands 13 and 14
OCSFN	+d	2	W	sfn of open instance for virtualization
	+d	1	S	Start/Stop console commands (98, 99)
CStpSKG	+d	1	B	Screen/Keyboard number
	+d	6	S	De-install driver command 20
DINEndLocn	+d	4	D	
DINLengthn	+11	2	W	

**PktStatus** word masks

Name	Bit Mask	Description
STERR	0x8000	Bit 15 - Error
STINTER	0x0400	Bit 10 - Interim character
STBUI	0x0200	Bit 9 - Busy
STDON	0x0100	Bit 8 - Done
STECODE	0x00ff	Error code
WRECODE	0	

#### PktFlag flags

Name	Flag value	Description
fPktInt13RP	0x01	Int 13 Request Packet
fPktCallOutDone	0x02	Int 13 Callout completed
fPktDiskIOTchd	0x04	Disk_IO has touched this packet
STDON	0x0100	Bit 8 - Done
STECODE	0x00ff	Error code
WRECODE	0	

See [Device Driver Strategy Commands](#) for a cross-reference of **PktCmd** command codes.

-----

## BIOS Parameter Block (BPB) for OS/2 Warp V4.0 and OS/2 Warp V3.0

#### Pointers

I/O Request Packet fields **InitpBPB** and **BldBPBbuffer** point to the BPB structure.

#### Locations

Allocated from the System Arena.

#### VM Owner

Non-specific.

#### Format

#### BPB

Field Name	Offset	Length	Type	Description
BPSECSZ	0	2	W	Size in bytes of physical sector
BPCLUS	2	1	B	Sectors/Alloc unit
BPRES	3	2	W	Number of reserved sectors
BPFTCNT	5	1	B	Number of FATs
BPRDCNT	6	2	W	Number of directory entries

BPSCCNT	8	2	W	Total number of sectors
BPMEDIA	a	1	B	Media descriptor byte
BPFTSEC	b	2	W	Number of sectors taken up by one FAT
BPBcSecPerTrack	d	2	W	sectors per track
BPBcHeads	f	2	W	number of heads
BPBcSecHidden	11	2	W	number of hidden sectors before the reserved sectors
BPBcSecHiddenH	13	2	W	High word of hidden sectors
BPBcSecTotal	15	4	D	Big total sectors (if BPSCCNT = 0)
PHYDRV	19	1	B	PHYSICAL DRIVE NUMBER (0 OR 80H)
CURHD	1a	1	B	Unitialized

-----

## Reference Tables

The following reference information is tabulated in this section:

[OS/2 System Error Codes](#)

[OS/2 System Exception Codes](#)

[Trap Screen Reference](#)

[Standard GDT Assignments](#)

[Standard LDT Assignments](#)

[VM System Owner Ids](#)

[DevHlp Function Cross-reference](#)

[Device Driver Strategy Commands](#)

[System Ordinal Cross-reference](#)

[OS/2 Fix Pack to Build Level Cross-reference](#)

-----

## System Error Codes

OS/2 System Error Codes

<i>Code</i>	<i>Description</i>
0	NO ERROR
1	INVALID FUNCTION
2	FILE NOT FOUND
3	PATH NOT FOUND

4	TOO MANY OPEN FILES
5	ACCESS DENIED
6	INVALID HANDLE
7	ARENA TRASHED
8	NOT ENOUGH MEMORY
9	INVALID BLOCK
10	BAD ENVIRONMENT
11	BAD FORMAT
12	INVALID ACCESS
13	INVALID DATA
15	INVALID DRIVE
16	CURRENT DIRECTORY
17	NOT SAME DEVICE
18	NO MORE FILES
19	WRITE PROTECT
20	BAD UNIT
21	NOT READY
22	BAD COMMAND
23	CRC
24	BAD LENGTH
25	SEEK
26	NOT DOS DISK
27	SECTOR NOT FOUND
28	OUT OF PAPER
29	WRITE FAULT
30	READ FAULT
31	GEN FAILURE
32	SHARING VIOLATION
33	LOCK VIOLATION
34	WRONG DISK
35	FCB UNAVAILABLE
36	SHARING BUFFER EXCEEDED
37	CODE PAGE MISMATCHED
38	HANDLE EOF
39	HANDLE DISK FULL
40	BAD COMMAND
41	CRC
42	BAD LENGTH
43	SEEK
44	NOT DOS DISK

45	SECTOR NOT FOUND
46	OUT OF PAPER
47	WRITE FAULT
48	READ FAULT
49	GEN FAILURE
50	NOT SUPPORTED
51	REM NOT LIST
52	DUP NAME
53	BAD NETPATH
54	NETWORK BUSY
55	DEV NOT EXIST
56	TOO MANY CMDS
57	ADAP HDW ERR
58	BAD NET RESP
59	UNEXP NET ERR
60	BAD REM ADAP
61	PRINTQ FULL
62	NO SPOOL SPACE
63	PRINT CANCELLED
64	NETNAME DELETED
65	NETWORK ACCESS DENIED
66	BAD DEV TYPE
67	BAD NET NAME
68	TOO MANY NAMES
69	TOO MANY SESS
70	SHARING PAUSED
71	REQ NOT ACCEP
72	REDIR PAUSED
73	SBCS ATT WRITE PROT
74	SBCS GENERAL FAILURE
75	XGA OUT MEMORY
80	FILE EXISTS
81	DUP FCB
82	CANNOT MAKE
83	FAIL I24
84	OUT OF STRUCTURES
85	ALREADY ASSIGNED
86	INVALID PASSWORD
87	INVALID PARAMETER



88	NET WRITE FAULT
89	NO PROC SLOTS
90	NOT FROZEN
NOT FROZEN	SYS COMP NOT LOADED
91	ERR TSTOVFL
92	ERR TSTDUP
93	NO ITEMS
95	INTERRUPT
96	INVALID DTA
99	DEVICE IN USE
100	TOO MANY SEMAPHORES
101	EXCL SEM ALREADY OWNED
102	SEM IS SET
103	TOO MANY SEM REQUESTS
104	INVALID AT INTERRUPT TIME
105	SEM OWNER DIED
106	SEM USER LIMIT
107	DISK CHANGE
108	DRIVE LOCKED
109	BROKEN PIPE
110	OPEN FAILED
111	BUFFER OVERFLOW
112	DISK FULL
113	NO MORE SEARCH HANDLES
114	INVALID TARGET HANDLE
115	PROTECTION VIOLATION
116	VIOKBD REQUEST
117	INVALID CATEGORY
118	INVALID VERIFY SWITCH
119	BAD DRIVER LEVEL
120	CALL NOT IMPLEMENTED
121	SEM TIMEOUT
122	INSUFFICIENT BUFFER
123	INVALID NAME
123	HPFS INVALID VOLUME CHAR
124	INVALID LEVEL
125	NO VOLUME LABEL
126	MOD NOT FOUND
127	PROC NOT FOUND

128	WAIT NO CHILDREN
129	CHILD NOT COMPLETE
130	DIRECT ACCESS HANDLE
131	NEGATIVE SEEK
132	SEEK ON DEVICE
133	IS JOIN TARGET
134	IS JOINED
135	IS SUBSTED
136	NOT JOINED
137	NOT SUBSTED
138	JOIN TO JOIN
139	SUBST TO SUBST
140	JOIN TO SUBST
141	SUBST TO JOIN
142	BUSY DRIVE
143	SAME DRIVE
144	DIR NOT ROOT
145	DIR NOT EMPTY
146	IS SUBST PATH
147	IS JOIN PATH
148	PATH BUSY
149	IS SUBST TARGET
150	SYSTEM TRACE
151	INVALID EVENT COUNT
152	TOO MANY MUXWAITERS
153	INVALID LIST FORMAT
154	LABEL TOO LONG
154	HPFS VOL LABEL LONG
155	TOO MANY TCBS
156	SIGNAL REFUSED
157	DISCARDED
158	NOT LOCKED
159	BAD THREADID ADDR
160	BAD ARGUMENTS
161	BAD PATHNAME
162	SIGNAL PENDING
163	UNCERTAIN MEDIA
164	MAX THRDS REACHED
165	MONITORS NOT SUPPORTED
166	UNC DRIVER NOT INSTALLED

167	LOCK FAILED
168	SWAPIO FAILED
169	SWAPIN FAILED
170	BUSY
171	INT TOO LONG
173	CANCEL VIOLATION
174	ATOMIC LOCK NOT SUPPORTED
175	READ LOCKS NOT SUPPORTED
180	INVALID SEGMENT NUMBER
181	INVALID CALLGATE
182	INVALID ORDINAL
183	ALREADY EXISTS
184	NO CHILD PROCESS
185	CHILD ALIVE NOWAIT
186	INVALID FLAG NUMBER
187	SEM NOT FOUND
188	INVALID STARTING CODESEG
189	INVALID STACKSEG
190	INVALID MODULETYPE
191	INVALID EXE SIGNATURE
192	EXE MARKED INVALID
193	BAD EXE FORMAT
194	ITERATED DATA EXCEEDS 64k
195	INVALID MINALLOCSIZE
196	DYNLINK FROM INVALID RING
197	IOPL NOT ENABLED
198	INVALID SEGDP
199	AUTODATASEG EXCEEDS 64k
200	RING2SEG MUST BE MOVABLE
201	RELOC CHAIN XEEDS SEGLIM
202	INFLOOP IN RELOC CHAIN
203	ENVVAR NOT FOUND
204	NOT CURRENT CTRY
205	NO SIGNAL SENT
206	FILENAME EXCED RANGE
207	RING2 STACK IN USE
208	META EXPANSION TOO LONG
209	INVALID SIGNAL NUMBER
210	THREAD 1 INACTIVE

211	INFO NOT AVAIL
212	LOCKED
213	BAD DYNALINK
214	TOO MANY MODULES
215	NESTING NOT ALLOWED
216	CANNOT SHRINK
217	ZOMBIE PROCESS
218	STACK IN HIGH MEMORY
219	INVALID EXITROUTINE RING
220	GETBUF FAILED
221	FLUSHBUF FAILED
222	TRANSFER TOO LONG
223	FORCENOSWAP FAILED
224	SMG NO TARGET WINDOW
228	NO CHILDREN
229	INVALID SCREEN GROUP
230	BAD PIPE
231	PIPE BUSY
232	NO DATA
233	PIPE NOT CONNECTED
234	MORE DATA
240	VC DISCONNECTED
250	CIRCULARITY REQUESTED
251	DIRECTORY IN CDS
252	INVALID FSD NAME
253	INVALID PATH
254	INVALID EA NAME
255	EA LIST INCONSISTENT
256	EA LIST TOO LONG
257	NO META MATCH
258	FINDNOTIFY TIMEOUT
259	NO MORE ITEMS
260	SEARCH STRUC REUSED
261	CHAR NOT FOUND
262	TOO MUCH STACK
263	INVALID ATTR
264	INVALID STARTING RING
265	INVALID DLL INIT RING
266	CANNOT COPY
267	DIRECTORY

268	OPLOCKED FILE
269	OPLOCK THREAD EXISTS
270	VOLUME CHANGED
271	FINDNOTIFY HANDLE IN USE
272	FINDNOTIFY HANDLE CLOSED
273	NOTIFY OBJECT REMOVED
274	ALREADY SHUTDOWN
275	EAS DIDNT FIT
276	EA FILE CORRUPT
277	EA TABLE FULL
278	INVALID EA HANDLE
279	NO CLUSTER
280	CREATE EA FILE
281	CANNOT OPEN EA FILE
282	EAS NOT SUPPORTED
283	NEED EAS FOUND
284	DUPLICATE HANDLE
285	DUPLICATE NAME
286	EMPTY MUXWAIT
287	MUTEX OWNED
288	NOT OWNER
289	PARAM TOO SMALL
290	TOO MANY HANDLES
291	TOO MANY OPENS
292	WRONG TYPE
293	UNUSED CODE
294	THREAD NOT TERMINATED
295	INIT ROUTINE FAILED
296	MODULE IN USE
297	NOT ENOUGH WATCHPOINTS
298	TOO MANY POSTS
299	ALREADY POSTED
300	ALREADY RESET
301	SEM BUSY
303	INVALID PROCID
304	INVALID PDELTA
305	NOT DESCENDANT
306	NOT SESSION MANAGER
307	INVALID PCLASS

308	INVALID SCOPE
309	INVALID THREADID
310	DOSSUB SHRINK
311	DOSSUB NOMEM
312	DOSSUB OVERLAP
313	DOSSUB BADSIZE
314	DOSSUB BADFLAG
315	DOSSUB BADSELECTOR
316	MR MSG TOO LONG
316	MGS MR MSG TOO LONG
317	MR MID NOT FOUND
318	MR UN ACC MSGF
319	MR INV MSGF FORMAT
320	MR INV IVCOUNT
321	MR UN PERFORM
322	TS WAKEUP
323	TS SEMHANDLE
324	TS NOTIMER
326	TS HANDLE
327	TS DATETIME
328	SYS INTERNAL
329	QUE CURRENT NAME
330	QUE PROC NOT OWNED
331	QUE PROC OWNED
332	QUE DUPLICATE
333	QUE ELEMENT NOT EXIST
334	QUE NO MEMORY
335	QUE INVALID NAME
336	QUE INVALID PRIORITY
337	QUE INVALID HANDLE
338	QUE LINK NOT FOUND
339	QUE MEMORY ERROR
340	QUE PREV AT END
341	QUE PROC NO ACCESS
342	QUE EMPTY
343	QUE NAME NOT EXIST
344	QUE NOT INITIALIZED
345	QUE UNABLE TO ACCESS
346	QUE UNABLE TO ADD
347	QUE UNABLE TO INIT

349	VIO INVALID MASK
350	VIO PTR
351	VIO APTR
352	VIO RPTR
353	VIO CPTR
354	VIO LPTR
355	VIO MODE
356	VIO WIDTH
357	VIO ATTR
358	VIO ROW
359	VIO COL
360	VIO TOPROW
361	VIO BOTROW
362	VIO RIGHTCOL
363	VIO LEFTCOL
364	SCS CALL
365	SCS VALUE
366	VIO WAIT FLAG
367	VIO UNLOCK
368	SGS NOT SESSION MGR
369	SMG INVALID SGID
369	SMG INVALID SESSION ID
370	SMG NOSG
370	SMG NO SESSIONS
371	SMG GRP NOT FOUND
371	SMG SESSION NOT FOUND
372	SMG SET TITLE
373	KBD PARAMETER
374	KBD NO DEVICE
375	KBD INVALID IOWAIT
376	KBD INVALID LENGTH
377	KBD INVALID ECHO MASK
377	KBD INVALID INPUT MASK
378	KBD INVALID INPUT MASK
379	MON INVALID PARMS
380	MON INVALID DEVNAME
381	MON INVALID HANDLE
382	MON BUFFER TOO SMALL
383	MON BUFFER EMPTY

384	MON DATA TOO LARGE
385	MOUSE NO DEVICE
386	MOUSE INV HANDLE
387	MOUSE INV PARMS
388	MOUSE CANT RESET
389	MOUSE DISPLAY PARMS
390	MOUSE INV MODULE
391	MOUSE INV ENTRY PT
392	MOUSE INV MASK
393	NO MOUSE NO DATA
394	NO MOUSE PTR DRAWN
395	INVALID FREQUENCY
396	NLS NO COUNTRY FILE
396	NO COUNTRY SYS
397	NLS OPEN FAILED
397	OPEN COUNTRY SYS
398	NLS NO CTRY CODE
398	NO COUNTRY OR CODEPAGE
399	NLS TABLE TRUNCATED
400	NLS BAD TYPE
401	NLS TYPE NOT FOUND
401	COUNTRY NO TYPE
402	VIO SMG ONLY
403	VIO INVALID ASCIIZ
404	VIO DEREGISTER
405	VIO NO POPUP
406	VIO EXISTING POPUP
407	KBD SMG ONLY
408	KBD INVALID ASCIIZ
409	KBD INVALID MASK
410	KBD REGISTER
411	KBD DEREGISTER
412	MOUSE SMG ONLY
413	MOUSE INVALID ASCIIZ
414	MOUSE INVALID MASK
415	MOUSE REGISTER
416	MOUSE DEREGISTER
417	SMG BAD ACTION
418	SMG INVALID CALL
419	SCS SG NOTFOUND



420 SCS NOT SHELL  
421 VIO INVALID PARMS  
422 VIO FUNCTION OWNED  
423 VIO RETURN  
424 SCS INVALID FUNCTION  
425 SCS NOT SESSION MGR  
426 VIO REGISTER  
427 VIO NO MODE THREAD  
428 VIO NO SAVE RESTORE THD  
429 VIO IN BG  
430 VIO ILLEGAL DURING POPUP  
431 SMG NOT BASESHELL  
432 SMG BAD STATUSREQ  
433 QUE INVALID WAIT  
434 VIO LOCK  
435 MOUSE INVALID IOWAIT  
436 VIO INVALID HANDLE  
437 VIO ILLEGAL DURING LOCK  
438 VIO INVALID LENGTH  
439 KBD INVALID HANDLE  
440 KBD NO MORE HANDLE  
441 KBD CANNOT CREATE KCB  
442 KBD CODEPAGE LOAD INCOMPL  
443 KBD INVALID CODEPAGE ID  
444 KBD NO CODEPAGE SUPPORT  
445 KBD FOCUS REQUIRED  
446 KBD FOCUS ALREADY ACTIVE  
447 KBD KEYBOARD BUSY  
448 KBD INVALID CODEPAGE  
449 KBD UNABLE TO FOCUS  
450 SMG SESSION NON SELECT  
451 SMG SESSION NOT FOREGRND  
452 SMG SESSION NOT PARENT  
453 SMG INVALID START MODE  
454 SMG INVALID RELATED OPT  
455 SMG INVALID BOND OPTION  
456 SMG INVALID SELECT OPT  
457 SMG START IN BACKGROUND  
458 SMG INVALID STOP OPTION

459 SMG BAD RESERVE  
460 SMG PROCESS NOT PARENT  
461 SMG INVALID DATA LENGTH  
462 SMG NOT BOUND  
463 SMG RETRY SUB ALLOC  
464 KBD DETACHED  
465 VIO DETACHED  
466 MOU DETACHED  
467 VIO FONT  
468 VIO USER FONT  
469 VIO BAD CP  
470 VIO NO CP  
471 VIO NA CP  
472 INVALID CODE PAGE  
473 CPLIST TOO SMALL  
474 CP NOT MOVED  
475 MODE SWITCH INIT  
476 CODE PAGE NOT FOUND  
477 UNEXPECTED SLOT RETURNED  
478 SMG INVALID TRACE OPTION  
479 VIO INTERNAL RESOURCE  
480 VIO SHELL INIT  
481 SMG NO HARD ERRORS  
482 CP SWITCH INCOMPLETE  
483 VIO TRANSPARENT POPUP  
484 CRITSEC OVERFLOW  
485 CRITSEC UNDERFLOW  
486 VIO BAD RESERVE  
487 INVALID ADDRESS  
488 ZERO SELECTORS REQUESTED  
489 NOT ENOUGH SELECTORS AVA  
490 INVALID SELECTOR  
491 SMG INVALID PROGRAM TYPE  
492 SMG INVALID PGM CONTROL  
493 SMG INVALID INHERIT OPT  
494 VIO EXTENDED SG  
495 VIO NOT PRES MGR SG  
496 VIO SHIELD OWNED  
497 VIO NO MORE HANDLES  
498 VIO SEE LOG

499	VIO ASSOCIATED DC
500	KBD NO CONSOLE
501	MOUSE NO CONSOLE
502	MOUSE INVALID HANDLE
503	SMG INVALID DEBUG PARMS
504	KBD EXTENDED SG
505	MOU EXTENDED SG
506	SMG INVALID ICON FILE
507	TRC PID NON EXISTENT
508	TRC COUNT ACTIVE
509	TRC SUSPENDED BY COUNT
510	TRC COUNT INACTIVE
511	TRC COUNT REACHED
512	NO MC TRACE
513	MC TRACE
514	TRC COUNT ZERO
515	SMG TOO MANY DDS
516	SMG INVALID NOTIFICATION
517	LF INVALID FUNCTION
518	LF NOT AVAIL
519	LF SUSPENDED
520	LF BUF TOO SMALL
521	LF BUFFER CORRUPTED
521	LF BUFFER FULL
522	LF INVALID DAEMON
522	LF INVALID RECORD
523	LF INVALID TEMPL
523	LF INVALID SERVICE
524	LF GENERAL FAILURE
525	LF INVALID ID
526	LF INVALID HANDLE
527	LF NO ID AVAIL
528	LF TEMPLATE AREA FULL
529	LF ID IN USE
530	MOU NOT INITIALIZED
531	MOUINITREAL DONE
532	DOSSUB CORRUPTED
533	MOUSE CALLER NOT SUBSYS
534	ARITHMETIC OVERFLOW

535	TMR NO DEVICE
536	TMR INVALID TIME
537	PVW INVALID ENTITY
538	PVW INVALID ENTITY TYPE
539	PVW INVALID SPEC
540	PVW INVALID RANGE TYPE
541	PVW INVALID COUNTER BLK
542	PVW INVALID TEXT BLK
543	PRF NOT INITIALIZED
544	PRF ALREADY INITIALIZED
545	PRF NOT STARTED
546	PRF ALREADY STARTED
547	PRF TIMER OUT OF RANGE
548	PRF TIMER RESET
549	HPFS CHKDSK NO PARM SPACE
550	HPFS CHKDSK NORECOGNIZE
551	HPFS CHKDSK NOROOT FIND
552	HPFS CHKDSK NOFIX FS ERROR
553	HPFS CHKDSK CORRECT FS ERR
554	HPFS CHKDSK ORGAN FIX
555	HPFS CHKDSK RELOC BBPDATA
556	HPFS CHKDSK REM CORRUM BLOC
557	HPFS CHKDSK REM CORRUP FIL
558	HPFS CHKDSK FIX SPACE ALLO
559	HPFS NOT FORMATTED DISK
560	HPFS CHKDSK COR ALLOC
561	HPFS CHKDSK SEARC UNALLOC
562	HPFS CHKDSK DET LOST DATA
563	HPFS CHKDSK PERCENT SEARC
564	HPFS CHKDSK LOST DATASEARC
565	HPFS CHKDSK CRIT NOREAD
566	HPFS CHKDSK DISK INUSE
567	HPFS CHKDSK RECOVTEMP RELOC
568	HPFS TOTAL DISK SPACE
569	HPFS DIR KBYTES
570	HPFS FILE KBYTES
571	HPFS KBYTES AVAILABLE
572	HPFS CHKDSK PLACE REC FILE
573	HPFS CHKDSK RECO DIR AS
574	HPFS CHKDSK PLACED DATA

575	HPFS CHKDSK RECOV EA
576	HPFS CHKDSK FIND EA INTERM
577	HPFS CHKDSK RELOC TEMP EA
578	HPFS CHKDSK RELOC AC LIST
579	HPFS CHKDSK LIST NORELOC
580	HPFS CHKDSK TRUN EA LIST
581	HPFS CHKDSK TRUN EA NAME
582	HPFS CHKDSK TRUN EA BBLOCK
583	HPFS CHKDSK REM INVALID EA
584	HPFS CHKDSK FIX EA ALLOC
585	HPFS CHKDSK FIX ALACCCTRL
586	HPFS CHKDSK ACCTR LIST BBL
587	HPFS CHKDSK REM ACLIST
588	HPFS CHKDSK FOUND DATANORL
589	HPFS WRONG VERSION
590	HPFS CHKDSK FOUND DATATEMP
591	HPFS CHKDSK FIX TEMPSTATUS
592	HPFS CHKDSK FIX NEEDEDATA
593	HPFS RECOVER PARM ERROR
594	HPFS RECOV FILE NOT FOUND
595	HPFS RECOV UNKNOWN ERROR
596	HPFS RECOV NOT ENOUGH MEM
597	HPFS RECOV NOWRITE DATA
598	HPFS RECOV NOTEMP CREATE
599	HPFS RECOV EA NOREAD
600	HPFS RECOV FILE BYTES
601	HPFS RECOV BAD BYTES RECOV
602	HPFS RECOV FILEBYTES NOREC
603	HPFS RECOV DISK INUSE
604	HPFS RECOV FILE NODELETE
605	HPFS RECOV NOCREATE NEWFILE
606	HPFS RECOV SYSTEM ERROR
607	HPFS SYS PARM ERROR
608	HPFS SYS CANNOT INSTALL
609	HPFS SYS DRIVE NOTFORMATED
610	HPFS SYS FILE NOCREATE
611	HPFS SIZE EXCEED
612	HPFS SYNTAX ERR
613	HPFS NOTENOUGH MEM

614	HPFS WANT MEM
615	HPFS GET RETURNED
616	HPFS SET RETURNED
617	HPFS BOTH RETURNED
618	HPFS STOP RETURNED
619	HPFS SETPRTYRETURNED
620	HPFS ALCSG RETURNED
621	HPFS MSEC SET
622	HPFS OPTIONS
623	HPFS POS NUM VALUE
624	HPFS VALUE TOO LARGE
625	HPFS LAZY NOT VALID
626	HPFS VOLUME ERROR
627	HPFS VOLUME DIRTY
628	HPFS NEW SECTOR
629	HPFS FORMAT PARM ERROR
630	HPFS CANNOT ACCESS CONFIG
631	HPFS RECOV FILE
632	HPFS CHKDSK KBYTES RESERVE
633	HPFS CHKDSK KBYTES IN EA
634	HPFS BYTEBUF SET
635	HPFS FORMATTING COMPLETE
636	HPFS WRONG VOLUME LABEL
637	HPFS FMAT TOO MANY DRS
638	VDD UNSUPPORTED ACCESS
639	VDD LOCK USEAGE DENIED
640	TIMEOUT
641	VDM DOWN
642	VDM LIMIT
643	VDD NOT FOUND
644	INVALID CALLER
645	PID MISMATCH
646	INVALID VDD HANDLE
647	VLPT NO SPOOLER
648	VCOM DEVICE BUSY
649	VLPT DEVICE BUSY
650	NESTING TOO DEEP
651	VDD MISSING
671	BIDI INVALID LENGTH
672	BIDI INVALID INCREMENT

673	BIDI INVALID COMBINATION
674	BIDI INVALID RESERVED
675	BIDI INVALID EFFECT
676	BIDI INVALID CSDREC
677	BIDI INVALID CSDSTATE
678	BIDI INVALID LEVEL
679	BIDI INVALID TYPE SUPPORT
680	BIDI INVALID ORIENTATION
681	BIDI INVALID NUM SHAPE
682	BIDI INVALID CSD
683	BIDI NO SUPPORT
684	NO BIDI RW INCOMPLETE
689	HPFS LAZY ON
690	HPFS LAZY OFF
691	IMP INVALID PARM
692	IMP INVALID LENGTH
693	MSG HPFS DISK WARN
694	MSG HPFS FNODE WARN
730	MON BAD BUFFER
731	MODULE CORRUPTED
732	BOOT DRIVE NOT ACCESSIBLE
1477	SM OUTOF SWAPFILE
2055	LF TIMEOUT
2057	LF SUSPEND SUCCESS
2058	LF RESUME SUCCESS
2059	LF REDIRECT SUCCESS
2060	LF REDIRECT FAILURE
32768	SWAPPER NOT ACTIVE
32769	INVALID SWAPID
32770	IOERR SWAP FILE
32771	SWAP TABLE FULL
32772	SWAP FILE FULL
32773	CANT INIT SWAPPER
32774	SWAPPER ALREADY INIT
32775	PMM INSUFFICIENT MEMORY
32776	PMM INVALID FLAGS
32777	PMM INVALID ADDRESS
32778	PMM LOCK FAILED
32779	PMM UNLOCK FAILED

32780	PMM MOVE INCOMPLETE
32781	UCOM DRIVE RENAMED
32782	UCOM FILENAME TRUNCATED
32783	UCOM BUFFER LENGTH
32784	MON CHAIN HANDLE
32785	MON NOT REGISTERED
32786	SMG ALREADY TOP
32787	PMM ARENA MODIFIED
32788	SMG PRINTER OPEN
32789	PMM SET FLAGS FAILED
32790	INVALID DOS DD
32791	BLOCKED
32792	NOBLOCK
32793	INSTANCE SHARED
32794	NO OBJECT
32795	PARTIAL ATTACH
32796	INCACHE
32797	SWAP IO PROBLEMS
32798	CROSSES OBJECT BOUNDARY
32799	LONGLOCK
32800	SHORTLOCK
32801	UVIRTLOCK
32802	ALIASLOCK
32803	ALIAS
32804	NO MORE HANDLES
32805	SCAN TERMINATED
32806	TERMINATOR NOT FOUND
32807	NOT DIRECT CHILD
32808	DELAY FREE
32809	GUARDPAGE
32900	SWAPERROR
32901	LDRError
32902	NOMEMORY
32903	NOACCESS
32904	NO DLL TERM
65026	CPSIO CODE PAGE INVALID
65027	CPSIO NO SPOOLER
65028	CPSIO FONT ID INVALID
65033	CPSIO INTERNAL ERROR
65034	CPSIO INVALID PTR NAME



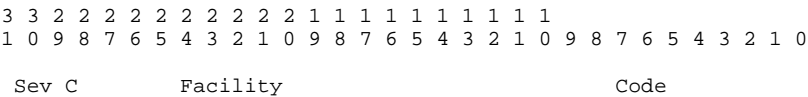
65037	CPSIO NOT ACTIVE
65039	CPSIO PID FULL
65040	CPSIO PID NOT FOUND
65043	CPSIO READ CTL SEQ
65045	CPSIO READ FNT DEF
65047	CPSIO WRITE ERROR
65048	CPSIO WRITE FULL ERROR
65049	CPSIO WRITE HANDLE BAD
65074	CPSIO SWIT LOAD
65077	CPSIO INV COMMAND
65078	CPSIO NO FONT SWIT
65079	ENTRY IS CALLGATE
0xFF00	USER DEFINED BASE

#### Dos INT 24 Critical Error Codes

<i>Code</i>	<i>Description</i>
0	I24 WRITE PROTECT
1	I24 BAD UNIT
2	I24 NOT READY
3	I24 BAD COMMAND
4	I24 CRC
5	I24 BAD LENGTH
6	I24 SEEK
7	I24 NOT DOS DISK
8	I24 SECTOR NOT FOUND
9	I24 OUT OF PAPER
10	I24 WRITE FAULT
11	I24 READ FAULT
12	I24 GEN FAILURE
13	I24 DISK CHANGE
15	I24 WRONG DISK
16	I24 UNCERTAIN MEDIA
17	I24 CHAR CALL INTERRUPTED
18	I24 NO MONITOR SUPPORT
19	I24 INVALID PARAMETER
20	I24 DEVICE IN USE
21	I24 QUIET INIT FAIL

# OS/2 System Exception Codes

Exception values are 32-bit values laid out as follows:



where

Sev - is the severity code  
00 - Success  
01 - Informational  
10 - Warning  
11 - Error

C - is the Customer code flag

Facility - is the facility code

Code - is the facility's status code

Exceptions specific to OS/2 2.0 (e.g. XCPT\_SIGNAL) will be marked with a facility code of 1.

80000001H	XCPT_GUARD_PAGE_VIOLATION		
	P1	Access Code	
		00000001H	XCPT_READ_ACCESS
		00000002H	XCPT_WRITE_ACCESS
	P2	FaultAddr	
80010001H	XCPT_UNABLE_TO_GROW_STACK		
0C0010001H	XCPT_PROCESS_TERMINATE		
	This exception is sent to a thread for Synchronous Process Termination and also for Thread Termination.		
0C0010002H	XCPT_ASYNC_PROCESS_TERMINATE		
	P1	TID of 'terminator' thread	
0C0010003H	XCPT_SIGNAL		
	P1	Signal Number	
		1	XCPT_SIGNAL_INTR
		3	XCPT_SIGNAL_KILLPROC
		4	XCPT_SIGNAL_BREAK
0C0010004H	XCPT_B1NPX_ERRATA_02		
0C0000005H	XCPT_ACCESS_VIOLATION		

This relates to Traps 0x09, 0x0b, 0x0c, 0x0d and 0x0e.

P1	Access Code	
	00000000H	XCPT_UNKNOWN_ACCESS
	00000001H	XCPT_READ_ACCESS
	00000002H	XCPT_WRITE_ACCESS
	00000004H	XCPT_EXECUTE_ACCESS
	00000008H	XCPT_SPACE_ACCESS
	00000010H	XCPT_LIMIT_ACCESS
P2		
	FaultAddr	XCPT_READ_ACCESS/XCPT_WRITE_ACCESS
	Selector	XCPT_SPACE_ACCESS
	-1	XCPT_LIMIT_ACCESS/XCPT_UNKNOWN_ACCESS

0C0000006H

XCPT\_IN\_PAGE\_ERROR

This relates to Trap 0x0e.

P1 FaultAddr

0C000001CH

XCPT\_ILLEGAL\_INSTRUCTION

This relates to Trap 0x06.

0C000001DH

XCPT\_INVALID\_LOCK\_SEQUENCE

0C0000024H

XCPT\_NONCONTINUABLE\_EXCEPTION

0C0000025H

XCPT\_INVALID\_DISPOSITION

0C0000026H

XCPT\_UNWIND

0C0000027H

XCPT\_BAD\_STACK

0C0000028H

XCPT\_INVALID\_UNWIND\_TARGET

0C0000093H

XCPT\_ARRAY\_BOUNDS\_EXCEEDED

This relates to Trap 0x05.

0C0000094H

XCPT\_FLOAT\_DENORMAL\_OPERAND

This relates to Trap 0x10.

0C0000095H

XCPT\_FLOAT\_DIVIDE\_BY\_ZERO

This relates to Trap 0x10.

0C0000096H

XCPT\_FLOAT\_INEXACT\_RESULT

This relates to Trap 0x10.

0C0000097H

XCPT\_FLOAT\_INVALID\_OPERATION

	This relates to Trap 0x10.		
0C0000098H	XCPT_FLOAT_OVERFLOW		
	This relates to Trap 0x10.		
0C0000099H	XCPT_FLOAT_STACK_CHECK		
	This relates to Trap 0x10.		
0C000009AH	XCPT_FLOAT_UNDERFLOW		
	This relates to Trap 0x10.		
0C000009BH	XCPT_INTEGER_DIVIDE_BY_ZERO		
	This relates to Trap 0x00.		
0C000009CH	XCPT_INTEGER_OVERFLOW		
	This relates to Trap 0x04.		
0C000009DH	XCPT_PRIVILEGED_INSTRUCTION		
	This relates to Trap 0x0d.		
0C000009EH	XCPT_DATATYPE_MISALIGNMENT		
	This relates to Trap 0x11.		
	P1	Access Code	
		00000001H	XCPT_READ_ACCESS
		00000002H	XCPT_WRITE_ACCESS
	P2	Alignment	
	P3	FaultAddr	
0C000009FH	XCPT_BREAKPOINT		
	This relates to Trap 0x03.		
0C00000A0H	XCPT_SINGLE_STEP		
	This relates to Trap 0x01.		

For a further information refer to:

- OS/2 Technical Library - Control Program Programming Reference, Appendix C.
- bsexcpt.h or bsexcpt.inc include files supplied with the OS/2 Programmers Toolkit.

-----

## Trap and Exeption Popup Message Reference

The trap screen and has in two basic formats:

The application exception (SYS0147, SYS317x, SYS3190) messages.

The Internal Processing Error (IPE).

### **Application Trap/Exception**

Application exception popups are logged in the POPUPLOG.OS2 file of which the following is an example. They are also displayed in a popup window in a slightly abbreviated form.

Control of exception logging and popup displays may be done from the **TRAPLOG** command or **SUPPRESSPOPUPS** CONFIG.SYS statement.

```
01-> 02-25-1999 10:58:35 SYS3175 PID 00b2 TID 0001 Slot 0068
02-> E:\CLASSES\LABS\LAB26\BEDBUG.EXE
03-> c0000005
04-> 1bf94e24
05-> P1=00000001 P2=00000000 P3=XXXXXXXX P4=XXXXXXXX
06-> EAX=00000000 EBX=00060210 ECX=0002881c EDX=00060210
07-> ESI=00000001 EDI=00000002
08-> DS=0053 DSACC=d0f3 DSLIM=1fffffffff
09-> ES=0053 ESACC=d0f3 ESLIM=1fffffffff
10-> FS=150b FSACC=00f3 FSLIM=00000030
11-> GS=0000 GSACC=**** GSLIM=*****
12-> CS:EIP=005b:1bf94e24 CSACC=d0df CSLIM=1fffffffff
13-> SS:ESP=0053:000287ec SSACC=d0f3 SSLIM=1fffffffff
14-> EBP=000287f8 FLG=00012206

15-> DOSCALL1.DLL 0002:00004e24
```

The information presented varies slightly according to circumstance. In general, inapplicable information is either omitted, or overlayed with asterisks (\*) or exes (X).

Each line of the trap screen conveys the following meaning:

1. Date and Time or Trap, Trap message Id and Failing Process Id, Thread Id and Thread Slot.
2. Failing process. In general this will not be the trapping module.
3. OS/2 System Exception code. See [OS/2 System Exception Codes](#) for a complete set of system generated exceptions.
4. 32-bit Instruction address at time of exception.
5. Exception Information Parameters. taken from the [Exception Report Record](#). See [OS/2 System Exception Codes](#) for the exception information parameters that are associated with each system exception.

#### **Note:**

When P1=00000000 and P2=FFFFFFFF, this frequently indicates that the trap occurred while executing a system API and that the previous instruction in the user's code was a call gate. When this happen a bad parameter has been passed to the system from the application.

If in addition the EFLAGS register has high word 0019xxxx then Virtual Machine Extensions may indirectly be causing a problem. Try VME=NO in CONFIG.SYS.

6. The EAX, EXB, EXC and EDX registers at the time the exception was reported.
7. The ESI and EDI registers at the time the exception was reported.
8. The DS selector at the time the exception was reported.

This information is presented in the form:

xS=nnnn	The selector value.
xSACC=nnnn	The descriptor access bits.

Reading from right to left the bits of the access field are assigned the following meaning:

0	(A) 1=Accessed
1	(W) 1=Writeable
2	(E) 1=Executable
3	0

4	(S) 1=Application 0=System
5 & 6	(DPL) Privilege Level
7	(P) 1=Segment present
8 - 11	0
12	(AVL) 1=UVIRT allocation
13	(D) 1=32-bit Operands/Data
14	0
15	(G) 1=4K granularity limit, 0=byte granularity limit

See the "INTEL Pentium User's Guide, Volume 3" for more information on descriptor formats.  
 xSLIM=nnnnnnnn The limit field from the descriptor.

9. The ES selector at the time the exception was reported.
10. The FS selector at the time the exception was reported.
11. The GS selector at the time the exception was reported.
12. The instruction address at the time the exception was reported, followed by the CS selector Limit and Access fields.
13. The stack address at the time the exception was reported, followed by the SS selector Limit and Access fields.
14. The EBP register and EFLAGS register.
15. The module name and relative object and offset within the module that corresponds to the exception address reported on line 4.

#### Notes:

Lines 3-5 are formatted from the [Exception Report Record](#) generated at the time the exception occurred.

Lines 6-14 are formatted from the [Exception Context Record](#) generated at the time the exception occurred.

Exception report and Context records can be modified by exception handlers so it is possible that the information displayed might not be correct.

Exception popups will not be generated if an exception handler attempts recovery by returning **XCPT\_CONTINUE\_EXECUTION** to the system. However this does not guarantee that the program will continue to operate correctly. When recovery is not successful it might be necessary to disable exception handlers. For Presentation Manager and the Workplace Shell this can be done by specifying **SET SHAPIEXCEPTIONHANDLER=OFF** and **SHELLEXCEPTIONHANDLER=OFF** in CONFIG.SYS. This can also be done from the kernel debugger by setting the first double-word of a thread's TIB to **0xffffffff**. An alternative approach is to use system trace to log exception handler dispatching. Minor codes 361, 362 and 363 of DOSCALL1 will log exception report and context records before and after they have been processed by an exception handler. Minor code 361 logs the original exception information, while minor code 363 logs the exception information that is returned to the system (and used for the popup log). Minor code 262 logs exception information before each exception handler is called. The [Exception Registration Record](#) is also logged by the tracepoint.

To activate these tracepoints issue the following command sequence:

```
TRACE ON /B:512 /D:PROCNAME,TID
TRACE ON DOSCALL1(361,362,363)
```

The first command defines a system trace buffer of 512Kb and turns on logging of Process name and Thread Id with each trace record. The second command activates the tracepoints. Use the **TRACEGET** and/or the **TRACEFMT** commands to extract and view the traced events.

Using trace to log exceptions will also provide full trap information for floating point exception, which is not provided by any other means.

The exception address reported in line 4 in most cases agrees with the **CS:EIP** reported in line 12. However, when a nested exception occurs, the register information will relate to the most recent exception. A particular example of this is where an Exit List Handler traps. Exit List Handlers are called when a process terminates, after any exception handling. The system first generates an **XCPT\_PROCESS\_TERMINATE** exception, which may be handled by exception handlers. If this exception is not recovered then process termination continues with Exit List processing. Once this starts, application exception handlers will not be called if any further exceptions are generated. If a further exception is generated then it will become a nested exception of the original **XCPT\_PROCESS\_TERMINATE**. If an Exit List Handler traps, a **SYS3170** popup will be generated with the register information in lines 6 - 14 corresponding to the nested exception and lines 3 - 5 and 15 corresponding to the original **XCPT\_PROCESS\_TERMINATE** exception.

If the system is unable to generate an exception popup message then a SYS0147 is generated. This can happen when there is insufficient kernel heap memory left to allocate a HARDERR request packet. This is not the only cause of a SYS0147, PM

resource (heap) shortages also cause this message. is exhausted this can occur.

SYS3190 occurs because of a TRAP 6. The application is incorrectly using LOCK prefixes either deliberately or possibly it had taken a wild jump to a non-instruction boundary.

### System Internal Processing Error (IPE)

The IPE message appears because of a fatal internal error condition. This may or may not be a trap, though the IPE trap is the most common.

The IPE message has the general format:

```
1-> <IPE specific Message>
2->  THE SYSTEM DETECTED AN INTERNAL PROCESSING
    ERROR AT LOCATION ##xxxx:yyyyyyyy - aaaa:bbbb
3->  11111 , ffff
4->  038600d1
5->  INTERNAL REVISION 6 . 307  DATE: 92/03/01
```

The parts of the IPE message are:

1. IPE specific message, which could be a simple line of text, for example:

CPS: Empty allocation block--not supported.

or a formatted register dump for a system trap, such as:

```
TRAP 0002          ERRCD= 0000  ERACC= ****  ERLIM= *****
EAX= 7d240a58  EBX= ff202fdc  ECX= 00064423  EDX= 00003624
ESI= fff3272c  EDI= 7d240004  EBP= 00004a44  FLG= 00003202
CS:EIP= 0160 : fff702a6  CSACC= c09d  CSLIM= ffffffff
SS:ESP= 0030 : 00004a38  SSACC= 1097  SSLIM= 00003fff
DS= 0158  DSACC= c0f3  DSLIM= ffffffff  CR0= ffffffff
ES= 0158  ESACC= c0f3  ESLIM= ffffffff  CR2= 1a060014
FS= 0000  FSACC= ****  FSLIM= *****
GS= 0000  GSACC= ****  GSLIM= *****
```

2. The CS:EIP of the caller to the kernel panic routine is shown as **##xxxx:yyyyyyyy**. For traps this will always be an address within the trap handler and not the address at which the error occurred - that is given in the error specific message.

The CS:EIP is prefixed with either **##** to indicate protect mode, paging enables in accordance with the Kernel Debugger command prompt.

The kernel relative object:offset address is shown as **aaaa:bbbb**.

3. **////** is intended to be the source line number at which the panic occurred. Values greater than 60000 are generated panic sequence numbers. **ffff** is the source file number.

**Note:** These values are mostly arbitrary and therefore not particularly useful. There is no published cross reference.

4. The processor ID.
5. The kernel revision information.

An example of the IPE trap screen is show in the following diagram:

```
1->  TRAP 0002          ERRCD= 0000  ERACC= ****  ERLIM= *****
    EAX= 7d240a58  EBX= ff202fdc  ECX= 00064423  EDX= 00003624
    ESI= fff3272c  EDI= 7d240004  EBP= 00004a44  FLG= 00003202
2->  CS:EIP= 0160 : fff702a6  CSACC= c09d  CSLIM= ffffffff
    SS:ESP= 0030 : 00004a38  SSACC= 1097  SSLIM= 00003fff
3->  DS= 0158  DSACC= c0f3  DSLIM= ffffffff  CR0= ffffffff
4->  ES= 0158  ESACC= c0f3  ESLIM= ffffffff  CR2= 1a060014
    FS= 0000  FSACC= ****  FSLIM= *****
    GS= 0000  GSACC= ****  GSLIM= *****
    THE SYSTEM DETECTED AN INTERNAL PROCESSING
```

ERROR AT LOCATION ##0160:fff6453f - 000d:a53f

60000 , 9084

038600d1

INTERNAL REVISION 6 . 307 DATE: 92/03/01

The register information may be interpreted as for application trap screens, with the following points notes:

1. This line shows the Trap number followed by the INTEL error code. Most often the associated error code is a selector number. When this is the case, this line formats the selector's access and limit values.
2. This line shows the address at which the trap occurred.
3. The value of control register 0 (CR0) is formatted after the DS register.  
CR0 contains processor control mode settings.
4. The value of control register 2 (CR2) is formatted after the ES register.  
CR2 contains the fault address for TRAP E errors.

## NMI Error Codes

NMI exceptions (TRAP 2) have no hardware defined Error Code. OS/2 uses the error code in the trap screen to indicate the source of the NMI as follows:

Error Code	Msg No	Description
0000	SYS1944	Software caused NMI (INT 2)
0001	SYS1945	RAM error, check memory (parity error)
0002	SYS1946	Adapter caused error (I/O channel check)
0003	SYS1947	Check bus mastering adapters, update adapter drivers. Also disable bus mastering on failing adapters as a problem determination tool to figure out which adapter is causing the failure. Contact the adapter vendor for further assistance. (DMA timeout)
0004	SYS1948	A device driver or Dos application disabled interrupts too long. Contact appropriate software vendor for updated software (Watchdog timeout).
0005	SYS3140	Contact application or device driver hardware vendor (software generated NMI).
0006	SYS3141	see error code 0003.
0007	SYS3142	see error code 0004.
0008	SYS3143	see error code 0002.
0009	SYS3149	see error code 0001.

## Standard GDT Assignments



The following table lists the GDT assignments that are statically assigned or assigned dynamically during initialisation.

This list is subject to change from release to release but may be verified by listing symbols from OS2KRNL segment DOSGDTDATA using the Kernel Debugger [LS command](#).

**Note:**

The Callgate descriptor assignments are shown for the **ALLSTRICT** kernel. For the **RETAIL** kernel they begin at one GTD entry earlier. Thus **GDT\_DOSALLOCSEG** through **GDT\_R0CSC** are assigned to selectors **1d08** through **1ea0**.

<i>Selector</i>	<i>Symbol</i>	<i>Description</i>
0	GDT	entry 0 is reserved (invalid)
8	GDT_GDT	entry 8 used to be GDT (now invalid)
10	GDT_TSS	Protect mode TSS
18	GDT_IDT	Protect Mode IDT
20	GDT_RM_IDT	Selector for 1st 1K
28	GDT_LDT	Selector for LDT
30	GDT_PTDA	PTDA/TCB/TSD selector
38	GDT_FPEM	Floating Point Emulator Work Area
40	GDT_ROMDATA	ROM data at physical address 400h
4a	GDT_R2DS	Ring 2 Data Selector
53	GDT_R3DS	Ring 3 Data Selector
5b	GDT_R3CS	Ring 3 Code Selector
63	GDT_R3PDS	Ring 3 Protected Data Selector
6b	GDT_R3THKDS,	Ring 3 Thunk Data Selector
70	GDT_SAS,	System Anchor Segment
78	GDT_DOSALIAS	SAS Read/Write Alias
80	GDT_SYSINFOSEG	InfosegGDT
88	GDT_DFTSS	Double Fault TSS
90	GDT_DFSTACK	Trap 8 stack selector
98	GDT_VPB	VPB BMP Segment
a0	GDT_RDR1	Reserved
a8	GDT_Buffers	Buffer Pool Segment
b0	GDT_Unused	unused selector (used to be MFT)
b8	GDT_RLR	RLR selector
c0	GDT_SFT	SFT selector of first SFT segment
c8	GDT_FSC	FSC array segment selector
d0	GDT_mFSD	mini-FSD
d8	GDT_RIPL	Remote IPL data
e0	GDT_NULLIDT	Invalid descriptor for mode switch
e8	GDT_INTSTACK	Interrupt stack alias

f0	GDT_RMCODE	386 modesw code selector
f8	GDT_RMDATA	386 modesw data selector
100	DOSHLP_CODESEL	DosHlp Code Selector
108	GDT_Pool	Start of dynamic GDT allocations
1508	GDT_Poolend	End of dynamic GDT allocations
150b	GDT_TIB	TIB selector
1d10	GDT_DOSALLOCSEG	DOSALLOCSEG call gate
1d18	GDT_DOSALLOCPROTSEG	DOSALLOCPROTSEG call gate
1d20	GDT_DOSDYNAMICTRACE	DOSDYNAMICTRACE call gate
1d28	GDT_DOSERROR	DOSERROR call gate
1d30	GDT_DOSFREERESOURCE	DOSFREERESOURCE call gate
1d38	GDT_DOSQUERYABIOSSUPPORT	DOSQUERYABIOSSUPPORT call gate
1d40	GDT_DOS16LDRDIRTYWORKER	DOS16LDRDIRTYWORKER call gate
1d48	GDT_DOSFREESEG	DOSFREESEG call gate
1d50	GDT_DOSGETPROCADDR	DOSGETPROCADDR call gate
1d58	GDT_DOSIEXECPGM	DOSIEXECPGM call gate
1d60	GDT_DOSIQAPPTYPE	DOSIQAPPTYPE call gate
1d68	GDT_DOSISEMWAIT	DOSISEMWAIT call gate
1d70	GDT_DOSLOADMODULE	DOSLOADMODULE call gate
1d78	GDT_DOSMAKEPIPE	DOSMAKEPIPE call gate
1d80	GDT_DOSREALLOCSEG	DOSREALLOCSEG call gate
1d88	GDT_DOSSICG	DOSSICG call gate
1d90	GDT_PANICWRITE	PANICWRITE call gate
1d98	GDT_DOSSETPRTY	DOSSETPRTY call gate
1da0	GDT_DOSLOGMODE	DOSLOGMODE call gate
1da8	GDT_DOSSETCP	DOSSETCP call gate
1db0	GDT_DOSGLOBALSEG	DOSGLOBALSEG call gate
1db8	GDT_DOSCREATETHREAD	DOSCREATETHREAD call gate
1dc0	GDT_DOSEXIT	DOSEXIT call gate
1dc8	GDT_DOSEXITLIST	DOSEXITLIST call gate
1dd0	GDT_DOSFREEMODULE	DOSFREEMODULE call gate
1dd8	GDT_DOSRESUMETHREAD	DOSRESUMETHREAD call gate
1de0	GDT_DOSSLEEP	DOSSLEEP call gate
1de8	GDT_DOSSUSPENDTHREAD	DOSSUSPENDTHREAD call gate
1df0	GDT_DOSLIBINIT	DOSLIBINIT call gate
1df8	GDT_REDIR	REDIR call gate
1e00	GDT_DOSCHGFILEPTR	DOSCHGFILEPTR call gate
1e08	GDT_DOSPROTECTCHGFILEPTR	DOSPROTECTCHGFILEPTR call gate
1e10	GDT_DOSCLOSE	DOSCLOSE call gate
1e18	GDT_DOSPROTECTCLOSE	DOSPROTECTCLOSE call gate

1e20	GDT_DOSDELETE	DOSDELETE call gate
1e28	GDT_DOSDEVICTL	DOSDEVICTL call gate
1e30	GDT_DOSDEVICTL2	DOSDEVICTL2 call gate
1e38	GDT_DOSDUPHANDLE	DOSDUPHANDLE call gate
1e40	GDT_DOSICOPY	DOSICOPY call gate
1e48	GDT_DOSIREAD	DOSIREAD call gate
1e50	GDT_DOSIPROTECTREAD	DOSIPROTECTREAD call gate
1e58	GDT_DOSISETRELMAXFH	DOSISETRELMAXFH call gate
1e60	GDT_DOSIWRITE	DOSIWRITE call gate
1e68	GDT_DOSIPROTECTWRITE	DOSIPROTECTWRITE call gate
1e70	GDT_DOSMOVE	DOSMOVE call gate
1e78	GDT_DOSOPEN	DOSOPEN call gate
1e88	GDT_MSSTACK	
1e90	GDT_OS2LDR	os2ldr's data
1e98	GDT_NWDTSS	NMI TSS
1ea0	GDT_NWDSTACK	NMI Stack Selector
1ea8	GDT_R0CSC	R0 Code Selector for Init DDs

-----

## Standard LDT Assignments

The following table lists the LDT assignments that are defined by the system.

<i>Selector</i>	<i>Description</i>
7	Read/Only access to the current LDT
dff7	Read/Only access to the current Global Information Segment.
dfff	Read/Only access to the current Local Information Segment and the current threads's Thread Local Memory Area.

-----

## VM System Object Owner IDs

*System Object Owner identifiers* are a reserved range of **hobs** used as labels for identifying generic types of memory object. They are not handles to [VMOBs](#), but are used in the **own** and **hmte** fields of some **VMOBs** and also in the **own** fields of some [Kernel Heap Block](#) headers.

### Note:

From Warp 3.0 fix pack 35 and Warp 4.0 GA, system object ids for file system and physical device driver owners are no longer used.

Instead the true module handle of the driver module is used.

The following table lists the system objects IDs are defined. The names shown are those displayed by the Kernel Debugger and Dump Formatter when formatting VMOB structures:

<i>Name</i>	<i>ID</i>	<i>Description</i>
lielist	0xff2d	LDR LieLists
demversion	0xff2e	DEM fake version entries
vmbmapd	0xff2f	VM Arena Bitmap Directory
npipenpn	0xff30	Named pipe NPN segment
npipenp	0xff31	Named pipe NP segment
reqpkttc	0xff32	DD TCB request packets
reqpkt2	0xff33	DD strat2 request packets
spldevrmp	0xff34	Spool Dev RMP segment
chardevrmp	0xff35	Char Dev RMP segment
syssemrmp	0xff36	System Semaphore RMP segment
romdata	0xff37	ROM data
libpath	0xff38	LDR LibPath
jfnflags	0xff39	JFN flags
jfntable	0xff3a	JFN table
ptouvirt	0xff3b	PhysToUVirt
tkr3stack	0xff3c	Ring 3 stack
tkr2stack	0xff3d	Ring 2 stack
tkenv	0xff3e	User Environment
tktib	0xff3f	Thread Information Block
reqpkt1	0xff40	DD strat1 request packets
allocphys	0xff41	Allocated via DevHlp AllocPhys
khbdon	0xff42	Unusable donated heap page owner
krhrwlm	0xff43	Resident R/W 1Meg mem heap owner
krhrolm	0xff44	Resident R/W 1Meg mem heap owner
mmph	0xff45	dekko mapped memory
pageio	0xff46	pageio per-swap-file save block
fsreclok	0xff47	record lock record owner
		File System Drivers
fsd1	0xff48	FSD 1
fsd2	0xff49	FSD 2
fsd3	0xff4a	FSD 3
fsd4	0xff4b	FSD 4
fsd5	0xff4c	FSD 5

fsd6	0xff4d	FSD 6
fsd7	0xff4e	FSD 7
fsd8	0xff4f	FSD 8 and subsequent
		Device Drivers
dd1	0xff50	device driver 1
dd2	0xff51	device driver 2
dd3	0xff52	device driver 3
dd4	0xff53	device driver 4
dd5	0xff54	device driver 5
dd6	0xff55	device driver 6
dd7	0xff56	device driver 7
dd8	0xff57	device driver 8
dd9	0xff58	device driver 9
dd10	0xff59	device driver 10
dd11	0xff5a	device driver 11
dd12	0xff5b	device driver 12
dd13	0xff5c	device driver 13
dd14	0xff5d	device driver 14
dd15	0xff5e	device driver 15
dd16	0xff5f	device driver 16 and subsequent
		Miscellaneous Owners
fsclmap	0xff60	cluster map owner
cdsrmp	0xff61	Current Directory Structure RMP seg
tom	0xff62	Timeout Manager
abios	0xff63	Advanced BIOS
cache	0xff64	cache
dbgdcdb	0xff65	DBG Debug Control Block
dbgkdb	0xff66	DBG Kernel Debug Block
dbgwpcb	0xff67	DBP Watch Point Control Block
demsft	0xff68	DEM SFT array (for FCBs)
demfonto	0xff69	DEM font offsets
demfont	0xff6a	DEM font data
devhlp	0xff6b	allocated via devhlp AllocPhys
discard	0xff6c	discardable, zero fill object
doshlp	0xff6d	DosHelp segment
dyndtgp	0xff6e	DYN tracepoint parm block
dyndto	0xff6f	dynamic tracepoint
dyndtot	0xff70	tmp dynamic trace info

dynmtel	0xff71	DYN MTE dynamic trace link
emalloc	0xff72	EM86 malloc()
emtss	0xff73	EM86 TSS
device	0xff74	installed device driver
infoseg	0xff75	infoseg (local or global)
initmsg	0xff76	INIT saved message
init	0xff77	generic init-time only
intdirq	0xff78	INT IRQ info
intstack	0xff79	interrupt stack
iopllist	0xff7a	List of modules with IOPL
kdbalias	0xff7b	Kernel debugger alias
kdbsym	0xff7c	Kernel debugger symbol
kmhook	0xff7d	KM hook info
ksem	0xff7e	KSEM semaphore
lbdd	0xff7f	loadable base device driver
lid	0xff80	ABIOS logical identifier
monitor	0xff81	monitor segment
mshare	0xff82	named-shared
mshrmp	0xff83	RMP having mshare records
nmi	0xff84	non maskable interrupt
npv	0xff85	287/387 save area
orphan	0xff86	orphaned segment
prof	0xff87	profile support
ptogdt	0xff88	Allocated via dh_allocateGDTSelector
ptovirt	0xff89	PhysToVirt
puse	0xff8a	Page Usage
pusetmp	0xff8b	tmp Page Usage
perfview	0xff8c	Perfview
qscache	0xff8d	QuerySysInfo cache
ras	0xff8e	RAS segment
resource	0xff8f	Resource BMP segment
syserv	0xff90	system service
timer	0xff91	timer services segment
traphe	0xff92	TRAP Hard Error
		File System Owners
fsbuf	0xff93	file system buffer
cdevtmp	0xff94	Char DEV TMP
fsc	0xff95	FSC segment
dpb	0xff96	DPB

eatmp	0xff97	fat EA TMP
fatsrch	0xff98	fat search segment
gnotify	0xff99	FindNotify global segment
pnotify	0xff9a	FindNotify private segment
fsh	0xff9b	installable file sys helper
ifs	0xff9c	installable file system
mfsd	0xff9d	mini file system
mft	0xff9e	master file table
npibuf	0xff9f	Named pipe I/O buffer segment
pipe	0xffa0	pipe
sft	0xffa1	system file table
vpb	0xffa2	volume parameter block
		Loader Owners
ldcache	0xffa3	Loader Instance Data Cache
ldrld	0xffa4	LDR Dynamic Load record
invalid	0xffa5	Cache being made
ldrmte	0xffa6	mte
ldrpath	0xffa7	LDR MTE path
ldrnres	0xffa8	LDR non-resident names
prot16	0xffa9	Protect 16 list
		Boot Loader and Kernel Owners
os2krnl	0xffaa	os2krnl load image
os2ldr	0xffab	os2ldr load image
ripl	0xffac	Remote IPL (remote boot)
		Page Manager Owners
pgalias	0xffad	Temporary page manager aliases
pgbuf	0xffae	PG loader and swapper buffer
pgcrpte	0xffaf	PG Compat. region page table
dbgalias	0xffb0	debugger alias pte
pgdir	0xffb1	PG Page directory
pgkstack	0xffb2	kernel stack region
pgvp	0xffb3	VP array
pgpf	0xffb4	PF array
pgprt	0xffb5	Page Range Table
pgsyspte	0xffb6	PG System page tables
		Selector Manager Owners
gdt	0xffb7	SEL GDT
selheap	0xffb8	Selector-mapped heap block
ldt	0xffb9	SEL LDT
lock	0xffba	SEL Lock

selnop	0xffbb	NO-OP Locks
seluvirt	0xffbc	SEL UVIRT mapping
		Semaphore Owners
semmisc	0xffbd	SEM Miscellaneous
semmuxq	0xffbe	SEM Mux Queue
semopenq	0xffbf	SEM Open Queue
semrec	0xffc0	SEM SemRecord
semstr	0xffc1	SEM string
semstruc	0xffc2	SEM Main structure
semtable	0xffc3	SEM Private/Shared table
		Swapper Owners
smdfh	0xffc4	SM Disk Frame Heap
smsfn	0xffc5	SM SFN array
smsf	0xffc6	SM Swap Frame
		Tasking Owners
tkextlst	0xffc7	TK Exit List record
tkkmreg	0xffc8	TK dispatch (KM) registers
tklibif	0xffc9	TK LibInit Free Notification record
tklibi	0xffca	TK LibInit record
ptda	0xffcb	TK PTDA
tcb	0xffcc	TK TCB
tsd	0xffcd	TK TSD
		VDD, VDH, VDM Owners
vddblkh	0xffce	VDD block header
vddblk	0xffcf	VDD memory block
vddcfstr	0xffd0	VDD config.sys string
vddctmp	0xffd1	VDD creation tmp allocation
vddep	0xffd2	VDD Entry Point
vddheaph	0xffd3	VDD heap header
vddheap	0xffd4	heap objects to load VDDs
vddhook	0xffd5	VDD hook
vddla	0xffd6	VDD Linear Arena header
vddlr	0xffd7	VDD Linear arena Record
vddmod	0xffd8	VDD module record
vddopen	0xffd9	open VDD record
vddpddep	0xffda	VDD PDD Entry Point
vddproc	0xffdb	VDD procedure record
vddstr	0xffdc	VDD string
vdhfhook	0xffdd	VDH fault hook



vdhallo	0xffde	VDH services resident memory
vdhswap	0xffdf	VDH services swappable memory
vdmalias	0xffe0	VDM Alias
		Virtual Memory Manager Owners
vmah	0xffe1	VM arena header
vmal	0xffe2	VM Alias Record
vmar	0xffe3	VM Arena Record
vmbmap	0xffe4	VM Location Bitmap
vmco	0xffe5	VM Context Record
vmdead	0xffe6	VM Dead Object
vmhsh	0xffe7	VM Location Hash Table
vmkrhb	0xffe8	VM *UNKNOWN* busy KRHB
vmkrhf	0xffe9	VM free KRHB
vmkrhl	0xffea	VM end KRHB
vmkrhro	0xffeb	VM Public Kernel Resident R/O Heap
vmkrhrw	0xffec	VM Public Kernel Resident R/W Heap
vmkshd	0xffed	VM Swappable Heap Descriptor
vmkshro	0xffee	VM Public Kernel Swappable R/O Heap
vmkshrw	0xffef	VM Public Kernel Swappable R/W Heap
vmlock	0xffff0	VM long term lock manager
vmob	0xffff1	VM Object Record
vmgs	0xffff2	VM Screen Group Switch record
vmbmp16	0xffff3	VM Temp buf (BMP16)
shrind	0xffff4	reserved for shared indicator
give	0xffff5	giveable segment
get	0xffff6	gettable segment
giveget	0xffff7	giveable and gettable segment
preload	0xffff8	Loader's preload object

-----

## DevHlp Function Cross-reference

The following table is a cross-reference for DevHlp function names with request code. The request code is loaded into the **DL** register before calling Device\_Help.

<i>Function Name</i>	<i>Code</i>	<i>Description</i>
----------------------	-------------	--------------------

DevHlp_SchedClock	0x0	Called each timer tick
DevHlp_DevDone	0x1	Device I/O complete
DevHlp_Yield	0x2	yield CPU if resched set
DevHlp_TCYield	0x3	yield to time critical task
DevHlp_ProcBlock	0x4	Block on event
DevHlp_ProcRun	0x5	Unblock process
DevHlp_SemRequest	0x6	claim a semaphore
DevHlp_SemClear	0x7	release a semaphore
DevHlp_SemHandle	0x8	obtain a semaphore handle
DevHlp_PushRequest	0x9	Push the request
DevHlp_PullRequest	0xA	Pull next request from Q
DevHlp_PullParticular	0xB	Pull a specific request
DevHlp_SortRequest	0xC	Push request in sorted order
DevHlp_AllocReqPacket	0xD	allocate request packet
DevHlp_FreeReqPacket	0xE	free request packet
DevHlp_QueueInit	0xF	Init/Clear char queue
DevHlp_QueueFlush	0x10	flush queue
DevHlp_QueueWrite	0x11	Put a char in the queue
DevHlp_QueueRead	0x12	Get a char from the queue
DevHlp_Lock	0x13	Lock segment
DevHlp_Unlock	0x14	Unlock segment
DevHlp_PhysToVirt	0x15	convert physical address to virtual
DevHlp_VirtToPhys	0x16	convert virtual address to physical
DevHlp_PhysToUVirt	0x17	convert physical to LDT
DevHlp_AllocPhys	0x18	allocate physical memory
DevHlp_FreePhys	0x19	free physical memory
DevHlp_SetROMVector	0x1A	set a ROM service routine vector
DevHlp_SetIRQ	0x1B	set an IRQ interrupt
DevHlp_UnSetIRQ	0x1C	unset an IRQ interrupt
DevHlp_SetTimer	0x1D	set timer request handler
DevHlp_ResetTimer	0x1E	unset timer request handler
DevHlp_MonitorCreate	0x1F	create a monitor
DevHlp_Register	0x20	install a monitor
DevHlp_DeRegister	0x21	remove a monitor
DevHlp_MonWrite	0x22	pass data records to monitor
DevHlp_MonFlush	0x23	remove all data from stream
DevHlp_GetDOSVar	0x24	Return pointer to DOS variable
DevHlp_SendEvent	0x25	an event occurred
DevHlp_ROMCritSection	0x26	ROM Critical Section
DevHlp_VerifyAccess	0x27	Verify access to memory

DevHlp_RAS	0x28	Put info in RAS trace buffer
DevHlp_ABIOGetParms	0x29	Get ABIO Calling Parms
DevHlp_AttachDD	0x2A	Attach to a device driver
DevHlp_InternalError	0x2B	Signal an internal error
DevHlp_ModifyPriority	0x2C	Undocumented (used by PM)
DevHlp_AllocGDTSelector	0x2D	Allocate GDT Selectors
DevHlp_PhysToGDTSelector	0x2E	Convert phys addr to GDT sel
DevHlp_RealToProt	0x2F	Change from real to protected mode
DevHlp_ProtToReal	0x30	Change from protected to real mode
DevHlp_EOI	0x31	Send EOI to PIC
DevHlp_UnPhysToVirt	0x32	mark completion of PhysToVirt
DevHlp_TickCount	0x33	modify timer
DevHlp_GetLIDEntry	0x34	Obtain Logical ID
DevHlp_FreeLIDEntry	0x35	Release Logical ID
DevHlp_ABIOCall	0x36	Call ABIO
DevHlp_ABIOCommonEntry	0x37	Invoke Common Entry Point
DevHlp_GetDeviceBlock	0x38	Get ABIO Device Block
DevHlp_RegisterStackUsag	0x3A	Register for stack usage
DevHlp_LogEntry	0x3B	Place data in log buffer
DevHlp_VideoPause	0x3C	Video pause on/off - D607
DevHlp_Save_Message	0x3D	Save msg in SysInit Message Table
DevHlp_SegRealloc	0x3E	Realloc DD protect mode segment
DevHlp_PutWaitingQueue	0x3F	Put I/O request on waiting queue
DevHlp_GetWaitingQueue	0x40	Get I/O request from waiting queue
DevHlp_PhysToSys	0x41	Address conversion for the AOX
DevHlp_PhysToSysHook	0x42	Address conversion for the AOX
DevHlp_RegisterDeviceClass	0x43	Register DC entry point
DevHlp_RegisterPDD	0x50	Register PDD entry point with VDM manager for later PDD-VDD communication
DevHlp_RegisterBeep	0x51	register PTD beep service entry point with kernel
DevHlp_Beep	0x52	preempt beep service via PTD
DevHlp_FreeGDTSelector	0x53	Free allocated GDT selector
DevHlp_PhysToGDTSel	0x54	Convert Phys Addr to GDT sel with given access
DevHlp_VMLock	0x55	Lock linear address range
DevHlp_VMUnlock	0x56	Unlock address range
DevHlp_VMAlloc	0x56	Allocate memory
DevHlp_VMFree	0x58	Free memory or mapping
DevHlp_VMProcessToGlobal	0x59	Create global mapping to process memory
DevHlp_VMGlobalToProcess	0x5A	Create process mapping to global memory

DevHlp_VirtToLin	0x5B	Convert virtual address to linear
DevHlp_LinToGDTSelector	0x5C	Convert linear address to virtual
DevHlp_GetDescInfo	0x5D	Return descriptor information
DevHlp_LinToPageList	0x5E	build pagelist array from lin addr
DevHlp_PageListToLin	0x5F	map page list array to lin addr
DevHlp_PageListToGDTSelector	0x60	map page list array to GDT sel.
DevHlp_RegisterTmrDD	0x61	Register TMR Device Driver.
DevHlp_RegisterPerfCtrs	0x62	Register device driver perf. ctrs (PVW).
DevHlp_AllocateCtxHook	0x63	Allocate a context hook
DevHlp_FreeCtxHook	0x64	Free a context hook
DevHlp_ArmCtxHook	0x65	Arm a context hook
DevHlp_VMSetMem	0x66	commit/decommit memory
DevHlp_OpenEventSem	0x67	open an event semaphore
DevHlp_CloseEventSem	0x68	close an event semaphore
DevHlp_PostEventSem	0x69	post an event semaphore
DevHlp_ResetEventSem	0x6A	reset an event semaphore
DevHlp_RegisterFreq	0x6B	register PTD freq service entry point with kernel
DevHlp_DynamicAPI	0x6C	add a dynamic API
DevHlp_ProcRun2	0x6D	Unblock process via procrun2
DevHlp_CreateInt13VDM	0x6E	Create Int13 VDM (Internal Only) OEMINT13
DevHlp_RegisterKrnExit	0x6F	Used to capture Kernel Exits F78693
DevHlp_PMPPostEventSem	0x70	PM Post Event Semaphore
DevHlp_AcquireSpinLock	0x71	acquire Spin Lock (SMP only)
DevHlp_ReleaseSpinLock	0x72	release Spin Lock (SMP only)
DevHlp_InitIntMouseCursorData	0x73	Initialize Mouse/Cursor Data (SMP only)
DevHlp_StartIntMouseCursor	0x74	Start Int Time Mouse/Cursor (SMP only)
DevHlp_EndIntMouseCursor	0x75	End Int Time Mouse/Cursor (SMP only)
DevHlp_Port_IO	0x76	Port I/O (SMP only)
DevHlp_SetIRQMask	0x77	Set/Unset an IRQ Mask (SMP only)
DevHlp_GetIRQMask	0x78	Retrieve an IRQ Mask state (SMP only)
DevHlp_CreateSpinLock	0x79	create Spin Lock (SMP only)
DevHlp_FreeSpinLock	0x7A	free Spin Lock (SMP only)
DevHlp_KillProc	0x7D	Kill Proc
DevHlp_QSysState	0x7E	Query System State
DevHlp_OpenFile	0x7F	Ring-0 File system Write
DevHlp_CloseFile	0x80	Ring-0 File system Seek
DevHlp_ReadFile	0x81	Ring-0 File system Read
DevHlp_ReadFileAt	0x82	File system Read at (seek)

## Device Driver Strategy Commands Cross-reference

The following table is a cross-reference for Device Driver strategy routine command codes. The command code is located in the **PktCmd** field of the [Strategy 1 Request Packet](#).

Name	Bit Mask	Description
CMDInit	0x0	INIT command
CMDMedChk	0x1	Media Check
CMDBldBPB	0x2	build BPB
CMDIOCTLR	0x3	reserved for 3.x compatability
CMDINPUT	0x4	read data from device
CMDNDR	0x5	non-destructive read
CMDInputS	0x6	input status
CMDInputF	0x7	input flush
CMDOUTPUT	0x8	write data to device
CMDOUTPUTV	0x9	write data and verify
CMDOutputS	0xa	output status
CMDOutputF	0xb	output flush
CMDIOCTLW	0xc	reserved for 3.x compatability
CMDOpen	0xd	device open
CMDClose	0xe	device close
CMDRemMed	0xf	is media removable
CMDGenIOCTL	0x10	Generic IOCTL
CMDResMed	0x11	reset media uncertain
CMDGetLogMap	0x12	
CMDSetLogMap	0x13	
CMDDeInstall	0x14	De-Install driver
	0x15	reserved
CMDPartfixeddisks	0x16	Partitionable Fixed Disks
CMDGetfd_logunitsmap	0x17	Get Fixed Disk/Logical Unit Map
CMDInputBypass	0x18	cache bypass read data
CMDOutputBypass	0x19	cache bypass write data
CMDOutputBypassV	0x1a	cache bypass write data and verify
CMDInitBase	0x1b	INIT command for base DDs
CMDShutdown	0x1c	
CMDGetDevSupport	0x1d	query for extended capability

	0x1e	reserved
CMDInitComplete	0x1f	Init complete for all DD's
CMDSaveRestore	0x20	
	0x21 - 0x60	reserved
CMDAddOnPrep	0x61	Prepare for add on
CMDStar	0x62	start console output
CMDStop	0x63	stop console output

-----

## System Ordinal Cross-reference

The following table is a cross-reference for System Entry points by Ordinal number.

<i>Ord</i>	<i>Entry Point</i>
1	DOSICREATETHREAD
2	DOSCWAIT
3	DOSENTERCRITSEC
4	DOSIEXECPGM
5	DOSEXIT
6	DOSEXITCRITSEC
7	DOSEXITLIST
8	DOSGETINFOSEG
9	DOSGETPRTY
10	DOSKILLPROCESS
11	DOSSETPRTY
12	DOSPTRACE
13	DOSHOLD SIGNAL
14	DOSSETSIGHANDLER
15	DOSFLAGPROCESS
16	DOSMAKEPIPE
17	DOSISYSSEMCLEAR
18	DOSISEMREQUEST
19	DOSISYSSEMSET
20	DOSSEMSETWAIT
21	DOSISEMWAIT
22	DOSMUXSEMWAIT
23	DOSCLOSESEM

24	DOSCREATESEM
25	DOSOPENSEM
26	DOSRESUMETHREAD
27	DOSSUSPENDTHREAD
28	DOSSETDATETIME
29	DOSTIMERASYNC
30	DOSTIMERSTART
31	DOSTIMERSTOP
32	DOSSLEEP
33	DOSGETDATETIME
34	DOSALLOCSEG
35	DOSALLOCshrSEG
36	DOSGETshrSEG
37	DOSGIVESEG
38	DOSREALLOCSEG
39	DOSFREESEG
40	DOSALLOCHUGE
41	DOSGETHUGESHIFT
42	DOSREALLOCHUGE
43	DOSCREATECSALIAS
44	DOSLOADMODULE
45	DOSGETPROCADDR
46	DOSFREEMODULE
47	DOSGETMODHANDLE
48	DOSGETMODNAME
49	DOSGETMACHINEMODE
50	DOSBEEP
51	DOSCLIACCESS
52	DOSDEVCONFIG
53	DOSDEVICTL
54	DOSSGSWITCH
55	DOSSGSWITCHME
56	DOSBUFRESET
57	DOSCHDIR
58	DOSCHGFILEPTR
59	DOSCLOSE
60	DOSDELETE
61	DOSDUPHANDLE
62	DOSFILELOCKS
63	DOSFINDCLOSE

64	DOSFINDFIRST
65	DOSFINDNEXT
66	DOSMKDIR
67	DOSMOVE
68	DOSNEWSIZE
69	DOSPORTACCESS
70	DOSOPEN
71	DOSQCURDIR
72	DOSQCURDISK
73	DOSQFHANDSTATE
74	DOSQFILEINFO
75	DOSQFILEMODE
76	DOSQFSINFO
77	DOSQHANDTYPE
78	DOSQVERIFY
79	DOSIREAD
80	DOSRMDIR
81	DOSSELECTDISK
82	DOSSETFHANDSTATE
83	DOSSETFILEINFO
84	DOSSETFILEMODE
85	DOSSETMAXFH
86	DOSSETVERIFY
87	DOSIWRITE
88	DOSSYSTEMSERVICE
89	DOSSETVEC
90	DOSSYSTRACE
91	DOSGETENV
92	DOSGETVERSION
93	DOSQTRACEINFO
94	DOSGETPID
95	DOSOPEN2
96	DOSLIBINIT
97	DOSSETFSINFO
98	DOSQPATHINFO
99	DOSDEVIOTL2
100	DOSICANONICALIZE
101	DOSSETFGND
102	DOSSWAPTASKINIT



103 DOSREADPHYS  
104 DOSSETPATHINFO  
105 DOSSGSWITCHPROC2  
106 STRUCHECK  
107 STRURESUPDATE  
108 DOSISETRELMAXFH  
109 DOSIDEVIOCTL  
110 DOS32FORCEDELETE  
111 DOS32KILLTHREAD  
112 DOSQUERYRASINFO  
113 DOS32DUMPPROCESS  
114 DOS32SUPPRESSPOPUPS  
118 DOSOPEN2COMPT  
119 DOSGETSTDA  
120 DOSERROR  
121 DOSGETSEG  
122 DOSLOCKSEG  
123 DOSUNLOCKSEG  
124 DOSSGSWITCHPROC  
125 DOSIRAMSEMWAKE  
126 DOSSIZESEG  
127 DOSMEMAVAIL  
128 DOSIRAMSEMREQUEST  
129 DOSPHYSICALDISK  
130 DOSGETCP  
131 DOSISETCP  
132 DOSGLOBALSEG  
133 DOSPROFILE  
134 DOSSEND SIGNAL  
135 DOSHUGESHIFT  
136 DOSHUGEINCR  
137 DOSREAD  
138 DOSWRITE  
139 DOSERRCLASS  
140 DOSSEMREQUEST  
141 DOSSEMCLEAR  
142 DOSSEMWAIT  
143 DOSSEMSET  
144 DOSEXECPGM  
145 DOSCREATETHREAD

146 DOSSUBSET  
147 DOSSUBALLOC  
148 DOSSUBFREE  
149 DOSREADASYNC  
150 DOSWRITEASYNC  
151 DOSSEARCHPATH  
152 DOSSCANENV  
153 DOSSETCP  
154 DOSQPROCSTATUS  
155 DOSGETRESOURCE  
156 DOSGETPPID  
157 DOSCALLBACK  
158 DOSICALLBACK  
159 DOSRETFORWARD  
160 DOSR2STACKREALLOC  
161 DOSFSRAMSEMREQUEST  
162 DOSFSRAMSEMCLEAR  
163 DOSQAPPTYPE  
164 DOSSETPROCCP  
165 DOSDYNAMICTRACE  
166 DOSQSYSINFO  
167 DOSIMAKENMPIPE  
168 DOSICALLNMPIPE  
169 DOSICONNECTNMPIPE  
170 DOSIDISCONNECTNMPIPE  
171 DOSIPEEKNMPIPE  
172 DOSIQNMPIPEINFO  
173 DOSIQNMPHANDSTATE  
174 DOSISETNMPHANDSTATE  
175 DOSITRANSACTNMPIPE  
176 DOSIWAITNMPIPE  
177 DOSISETNMPIPESEM  
178 DOSIQNMPIPESEMSTATE  
179 DOSIRAWREADNMPIPE  
180 DOSIRAWWRITENMPIPE  
181 DOSFSATTACH  
182 DOSQFSATTACH  
183 DOSFSCTL  
184 DOSFINDFIRST2

185 DOSMKDIR2  
186 DOSFILEIO  
187 DOSFINDNOTIFYCLOSE  
188 DOSFINDNOTIFYFIRST  
189 DOSFINDNOTIFYNEXT  
190 DOSSETTRACEINFO  
191 DOSEEDITNAME  
192 DOSLOGMODE  
193 DOSLOGENTRY  
194 DOSGETLOGBUFFER  
195 DOSLOGREGISTER  
196 DOSLOGREAD  
197 DOSFINDFROMNAME  
198 DOSOPLOCKRELEASE  
199 DOSOPLOCKWAIT  
200 DOSICOPY  
201 DOSCOPY  
202 DOSIQAPPTYPE  
203 DOSFORCEDELETE  
204 DOSENUMATTRIBUTE  
205 DOSOPLOCKSHUTDOWN  
206 DOSSHUTDOWN  
207 DOSGETRESOURCE2  
208 DOSFREERESOURCE  
209 DOS32SETMAXFH  
210 DOS32SETVERIFY  
211 DOS32ERRCLASS  
212 DOS32ERROR  
213 DOSCREATEVDM  
214 DOSMAXPATHLEN  
215 DOSPAGESIZE  
216 DOSLOCALINFO  
217 DOSGLOBALINFO  
218 DOS32SETFILEINFO  
219 DOS32SETPATHINFO  
220 DOS32SETDEFAULTDISK  
221 DOS32SETFHSTATE  
222 DOS32SETFSINFO  
223 DOS32QUERYPATHINFO  
224 DOS32QUERYHTYPE

225 DOS32QUERYVERIFY  
226 DOS32DELETEDIR  
227 DOS32SCANENV  
228 DOS32SEARCHPATH  
229 DOS32SLEEP  
230 DOS32GETDATETIME  
231 DOS32DEVCONFIG  
232 DOS32ENTERCRITSEC  
233 DOS32EXITCRITSEC  
234 DOS32EXIT  
235 DOS32KILLPROCESS  
236 DOS32SETPRIORITY  
237 DOS32RESUMETHREAD  
238 DOS32SUSPENDTHREAD  
239 DOS32CREATEPIPE  
240 DOS32CALLNPIPE  
241 DOS32CONNECTNPIPE  
242 DOS32DISCONNECTNPIPE  
243 DOS32CREATENPIPE  
244 DOS32PEEKNPIPE  
245 DOS32QUERYNPSTATE  
246 DOS32RAWREADNPIPE  
247 DOS32RAWWRITENPIPE  
248 DOS32QUERYNPPIPEINFO  
249 DOS32QUERYNPPIPESEMSTATE  
250 DOS32SETNPSTATE  
251 DOS32SETNPPIPESEM  
252 DOS32TRANSACTNPIPE  
253 DOS32WAITNPIPE  
254 DOS32RESETBUFFER  
255 DOS32SETCURRENTDIR  
256 DOS32SETFILEPTR  
257 DOS32CLOSE  
258 DOS32COPY  
259 DOS32DELETE  
260 DOS32DUPHANDLE  
261 DOS32EDITNAME  
263 DOS32FINDCLOSE  
264 DOS32FINDFIRST

265 DOS32FINDNEXT  
266 DOSOPENVDD  
267 DOSREQUESTVDD  
268 DOSCLOSEVDD  
269 DOS32FSATTACH  
270 DOS32CREATEDIR  
271 DOS32MOVE  
272 DOS32SETFILESIZE  
273 DOS32OPEN  
274 DOS32QUERYCURRENTDIR  
275 DOS32QUERYCURRENTDISK  
276 DOS32QUERYFHSTATE  
277 DOS32QUERYFSATTACH  
278 DOS32QUERYFSINFO  
279 DOS32QUERYFILEINFO  
280 DOS32WAITCHILD  
281 DOS32READ  
282 DOS32WRITE  
283 DOS32EXECPGM  
284 DOS32DEVIOCTL  
285 DOS32FSCTL  
286 DOS32BEEP  
287 DOS32PHYSICALDISK  
288 DOS32SETCP  
289 DOS32SETPROCESSCP  
290 DOS32STOPTIMER  
291 DOS32QUERYCP  
292 DOS32SETDATETIME  
293 THK32ALLOCBLOCK  
294 THK32FREEBLOCK  
295 THK32R3DS  
296 DOS32EXITLIST  
297 DOS32ALLOCPROTECTEDMEM  
298 DOS32ALIASMEM  
299 DOS32ALLOCMEM  
300 DOS32ALLOCSHAREDMEM  
301 DOS32GETNAMEDSHAREDMEM  
302 DOS32GETSHAREDMEM  
303 DOS32GIVESHAREDMEM  
304 DOS32FREEMEM

305 DOS32SETMEM  
306 DOS32QUERYMEM  
307 DOS32QUERYMEMSTATE  
308 DOS32OPENVDD  
309 DOS32REQUESTVDD  
310 DOS32CLOSEVDD  
311 DOS32CREATETHREAD  
312 DOS32GETINFOBLOCKS  
313 DOSALLOCPROTSEG  
314 DOSALLOCSHRPROTSEG  
315 DOSALLOCPROTHUGE  
316 DOS32DYNAMICTRACE  
317 DOS32DEBUG  
318 DOS32LOADMODULE  
319 DOS32QUERYMODULEHANDLE  
320 DOS32QUERYMODULENAME  
321 DOS32QUERYPROCADDR  
322 DOS32FREEMODULE  
323 DOS32QUERYAPPTYPE  
324 DOS32CREATEEVENTSEM  
325 DOS32OPENEVENTSEM  
326 DOS32CLOSEEVENTSEM  
327 DOS32RESETEVENTSEM  
328 DOS32POSTEVENTSEM  
329 DOS32WAITEVENTSEM  
330 DOS32QUERYEVENTSEM  
331 DOS32CREATEMUTEXSEM  
332 DOS32OPENMUTEXSEM  
333 DOS32CLOSEMUTEXSEM  
334 DOS32REQUESTMUTEXSEM  
335 DOS32RELEASEMUTEXSEM  
336 DOS32QUERYMUTEXSEM  
337 DOS32CREATEMUXWAITSEM  
338 DOS32OPENMUXWAITSEM  
339 DOS32CLOSEMUXWAITSEM  
340 DOS32WAITMUXWAITSEM  
341 DOS32ADDMUXWAITSEM  
342 DOS32DELETEMUXWAITSEM  
343 DOS32QUERYMUXWAITSEM

344 DOS32SUBSETMEM  
345 DOS32SUBALLOCMEM  
346 DOS32SUBFREEMEM  
347 DOS32SUBUNSETMEM  
348 DOS32QUERYSYSINFO  
349 DOS32WAITTHREAD  
350 DOS32ASYNCTIMER  
351 DOS32STARTTIMER  
352 DOS32GETRESOURCE  
353 DOS32FREERESOURCE  
354 DOS32SETEXCEPTIONHANDLER  
355 DOS32UNSETEXCEPTIONHANDLER  
356 DOS32RAISEEXCEPTION  
357 DOS32UNWINDEXCEPTION  
358 DOS32QUERYPAGEUSAGE  
359 DOSQUERYMODFROMCS  
360 DOS32QUERYMODFROMEIP  
361 DOSFPDATAAREA  
362 DOS32TMRQUERYFREQ  
363 DOS32TMRQUERYTIME  
364 DOS32ALIASPERFCTRS  
365 DOS32CONFIGUREPERF  
366 DOS32DECONPERF  
367 DOS32REGISTERPERFCTRS  
368 DOS32QUERYSYSSTATE  
369 DOS32FLATCS  
370 DOS32FLATDS  
371 DOS32QUERYABIOSSUPPORT  
372 DOS32ENUMATTRIBUTE  
373 DOS32QUERYDOSPROPERTY  
374 DOS32SETDOSPROPERTY  
375 DOSQUERYDOSPROPERTY  
376 DOSSETDOSPROPERTY  
377 DOS32PROFILE  
378 DOS32SETSIGNALEXCEPTIONFOC  
379 DOS32SENDSIGNALEXCEPTION  
380 DOS32ENTERMUSTCOMPLETE  
381 DOS32EXITMUSTCOMPLETE  
382 DOS32SETRELMAXFH  
383 MSGPUTMESSAGE

384 MSGTRUEGETMESSAGE  
385 MSGINSMESSAGE  
386 MSG32INSERTMESSAGE  
387 MSG32PUTMESSAGE  
388 MSG32TRUEGETMESSAGE  
389 MSGIQUERYMESSAGECP  
390 MSG32IQUERYMESSAGECP  
391 NLSCASEMAP  
392 NLSGETCOLLATE  
393 NLSGETCTRYINFO  
394 NLSGETDBCSEV  
395 NLS32QUERYCTRYINFO  
396 NLS32QUERYDBCSENV  
397 NLS32MAPCASE  
398 NLS32QUERYCOLLATE  
399 NPIPEMAKENMPIPE  
400 NPIPEQNMPIPEINFO  
401 NPIPECONNECTNMPIPE  
402 NPIPEDISCONNECTNMPIPE  
403 NPIPEQNMPPHANDSTATE  
404 NPIPESETNMPPHANDSTATE  
405 NPIPEPEEKNMPIPE  
406 NPIPEWAITNMPIPE  
407 NPIPETRANSACTNMPIPE  
408 NPIPECALLNMPIPE  
409 NPIPERAWREADNMPIPE  
410 NPIPERAWWRITENMPIPE  
411 NPIPESETNMPPIPESEM  
412 NPIPEQNMPIPESEMSTATE  
413 HPFSSTARTLAZYWRITER  
414 QUEINSTANCEDATA  
415 DOS32SHUTDOWN  
416 DOS32ICACHEMODULE  
417 DOS32REPLACEMODULE  
418 DOS32ACKNOWLEDGESIGNALEXC  
419 DOS32TIB  
420 DOSTMRQUERYFREQ  
421 DOSTMRQUERYTIME  
422 DOSREGISTERPERFCTRS



423 DOSFLATTOSEL  
424 DOSSELTOFLAT  
425 DOS32FLATTOSEL  
426 DOS32SELTOFLAT  
427 DOSIODELAYCNT  
428 DOS32SETFILELOCKS  
429 DOS32CANCELLOCKREQUEST  
430 LOGOPEN  
431 LOGCLOSE  
432 LOGADDENTRIES  
433 LOGGETENTRIES  
434 LOGSETSTATE  
435 LOGSETNAME  
436 LOGQUERYSTATE  
437 DOSOPENCHANGENOTIFY  
438 DOSRESETCHANGENOTIFY  
439 DOSCLOSECHANGENOTIFY  
440 DOS32OPENCHANGENOTIFY  
441 DOS32RESETCHANGENOTIFY  
442 DOS32CLOSECHANGENOTIFY  
443 DOSQUERYABIOSSUPPORT  
444 DOS32FORCESYSTEMDUMP  
454 DOS32ALLOCTHREADLOCALMEMORY  
455 DOS32FREETHREADLOCALMEMORY  
460 DOS32VERIFYFPIDTID  
464 PTDA\_LANMAN\_SEC  
465 PTDA\_PID  
466 SAS\_SEL  
467 TCB\_OPCOOKIE  
468 TCB\_OPFLAGS  
469 TCB\_NEWFLAGS  
470 TCB\_USER\_ID  
471 TCB\_PROC\_ID  
472 TCB\_FSHARING  
473 TCB\_SRVATTRIB  
474 TCB\_ALLOWED  
475 TCB\_PRTCB  
476 TCB\_NUMBER  
477 TCB\_THISSFT  
478 TCB\_THISCDS

479 TKOPTDA  
480 PTDA\_CRITSEC  
481 PTDA\_HOLD SIGCNT  
482 PTDA\_PPTDAPARENT  
483 PTDA\_PGDATA  
484 PTDA\_HANDLE  
485 PTDA\_MODULE  
486 PTDA\_LDTHANDLE  
487 PTDA\_CODEPAGE\_TAG  
488 PTDA\_JFN\_LENGTH  
489 PTDA\_JFN\_PTABLE  
490 PTDA\_JFN\_FLG\_PTR  
491 PTDA\_EXTERR\_LOCUS  
492 PTDA\_EXTERR  
493 PTDA\_EXTERR\_ACTION  
494 PTDA\_EXTERR\_CLASS  
495 PTDA\_PPID  
496 PTDA\_PROCTYPE  
497 PTDA\_CURRTCB  
498 PTDA\_CURRTSD  
499 PTDA\_SIGNATURE  
545 DOS32EXCEPTIONCALLBACK  
548 DOS32R3EXCEPTIONDISPATCHER  
549 DOSLIBIDISP  
550 DOSLIBIDISP16  
551 DOSLIBIDISP32  
552 DOSR3EXITADDR  
553 DOS32R3EXITADDR  
554 DOS32IREAD  
556 DOS32IWRITE  
565 DOSSETFILEINFO  
566 DOSSETPATHINFO  
569 DOSIFINDNEXT  
572 DOS32QUERYRESOURCE SIZE  
573 DOSQUERYRESOURCE SIZE  
574 DOSPMSEMWAIT  
575 DOSPMUXSEMWAIT  
576 THK16\_UNITHUNK  
577 HT16\_STARTUP

580 DOS32INITIALIZEPORTHOLE  
582 DOS32QUERYHEADERINFO  
583 DOSINITIALIZEPORTHOLE  
584 DOSQUERYHEADERINFO  
585 MON32MONREAD  
586 DOS32QUERYPROCTYPE  
587 DOSQUERYPROCTYPE  
588 MON32MONWRITE  
589 DOSISIGDISPATCH  
592 DOS32DLLTERMDISP  
594 DOS32IRAISEEXCEPTION  
597 DOS32IQUERYFHSTATE  
598 DOS32ISETFHSTATE  
599 DOSLDTSEL  
600 DOS32R3FRESTOR  
601 DOSIFINDFIRST  
615 DOS32IPROTECTWRITE  
617 DOSIPROTECTSETFILEINFO  
618 DOS32IPROTECTSETFILEINFO  
619 DOS32IPROTECTSETFHSTATE  
620 DOS32IPROTECTQUERYFHSTATE  
621 DOS32PROTECTSETFILEPTR  
622 DOSPROTECTCLOSE  
623 DOSPROTECTFILEIO  
624 DOSPROTECTFILELOCKS  
625 DOSIPROTECTREAD  
626 DOSIPROTECTWRITE  
627 DOSPROTECTNEWSIZE  
628 DOSPROTECTOPEN  
629 DOSPROTECTQFHANDSTATE  
630 DOSPROTECTSETFHANDSTATE  
631 DOSPROTECTQFILEINFO  
632 DOSPROTECTSETFILEINFO  
634 DOSPROTECTCHGFILEPTR  
635 DOSPROTECTENUMATTRIBUTE  
636 DOS32PROTECTENUMATTRIBUTE  
637 DOS32PROTECTOPEN  
638 DOS32PROTECTCLOSE  
639 DOS32PROTECTSETFILELOCKS  
640 DOS32PROTECTSETFILESIZE

641 DOS32PROTECTREAD  
642 DOS32PROTECTWRITE  
643 DOS32PROTECTSETFILEINFO  
644 DOS32PROTECTSETFHSTATE  
645 DOS32PROTECTQUERYFHSTATE  
646 DOS32PROTECTQUERYFILEINFO  
647 DOS32IPROTECTREAD  
649 MSGCLOSEMESSAGEFILE  
650 DOSLDRDIRTYWORKER  
651 DOS16LDRDIRTYWORKER  
661 QUEDOS32READQUEUE  
662 QUEDOS32PURGEQUEUE  
663 QUEDOS32CLOSEQUEUE  
664 QUEDOS32QUERYQUEUE  
665 QUEDOS32PEEKQUEUE  
666 QUEDOS32WRITEQUEUE  
667 QUEDOS32OPENQUEUE  
668 QUEDOS32CREATEQUEUE  
669 SMGDOS32STARTSESSION  
670 SMGDOS32SELECTSESSION  
671 SMGDOS32SETSESSION  
672 SMGDOS32STOPSESSION  
673 SMGREGISTERNOTIFICATION  
674 QUEDOSREADQUEUE  
675 QUEDOSPURGEQUEUE  
676 QUEDOSCLOSEQUEUE  
677 QUEDOSQUERYQUEUE  
678 QUEDOSPEEKQUEUE  
679 QUEDOSWRITEQUEUE  
680 QUEDOSOPENQUEUE  
681 QUEDOSCREATEQUEUE  
682 CHRDOSSMGETME  
683 CHRDOSSMFREEMEM  
684 CHRDOSSMGETSGCB  
685 CHRDOSSMINITSGCB  
686 SMGDOSMSGDOPOPUP  
687 SMGDOSMSWITCH  
688 SMGDOSMSERVEAPPREQ  
689 SMGDOSGETTIMES

690 SMGDOSSMSETTITLE  
691 SMGDOSSCRUNLOCK  
692 SMGDOSSMDOAPPREQ  
693 SMGDOSSTOPSESSION  
694 SMGDOSSELECTSESSION  
695 SMGDOSSCRLOCK  
696 SMGDOSSAVREDRAWWAIT  
697 SMGDOSSAVREDRAWUNDO  
698 SMGDOSSMSGENDPOPUP  
699 SMGDOSSETSESSION  
700 SMGDOSSETMNLCKTIME  
701 SMGDOSMODEUNDO  
702 SMGDOSSTARTSESSION  
703 SMGDOSSMGETSTATUS  
704 SMGDOSSMMODEWAIT  
705 SMGDOSSMTERMINATE  
706 SMGDOSSMGETAPPREQ  
707 SMGDOSSMINIALIZE  
708 SMGDOSSMSTART  
709 SMGDOSSMPARENTSWITCH  
710 SMGDOSSMPAUSE  
711 SMGDOSSMHDEINIT  
712 SMGDOSSMPMPRESENT  
713 SMGDOSSMREGISTERDD  
714 SMGDOSSMNOTIFYDD  
715 SMGDOSSMNOTIFYDD2  
716 SMGDOSSMOPENDD  
717 SMGDOSSMSETSESSIONTYPE  
718 CHRBASEINIT  
719 MOUDOSGETPTRSHAPE  
720 MOUDOSSETPTRSHAPE  
721 MOUDOSGETNUMMICKEYS  
722 MOUDOSGETTHRESHOLD  
723 MOUDOSSHELLINIT  
724 MOUDOSGETSCALEFACT  
725 MOUDOSFLUSHQUE  
726 MOUDOSGETNUMBUTTONS  
727 MOUDOSCLOSE  
728 MOUDOSSETTHRESHOLD  
729 MOUDOSSETSCALEFACT

730 MOUDOSGETNUMQUEEL  
731 MOUDOSDEREGISTER  
732 MOUDOSGETEVENTMASK  
733 MOUDOSSETEVENTMASK  
734 MOUDOSOPEN  
735 MOUDOSREMOVEPTR  
736 MOUDOSGETPTRPOS  
737 MOUDOSREADEVENTQUE  
738 MOUDOSSETPTRPOS  
739 MOUDOSGETDEVSTATUS  
740 MOUDOSSYNCH  
741 MOUDOSREGISTER  
742 MOUDOSSETDEVSTATUS  
743 MOUDOSDRAWPTR  
744 MOUDOSINITREAL  
745 KBDDOSSETCUSTXT  
746 KBDDOSPROCESSINIT  
747 KBDDOSGETCP  
748 KBDDOSCHARIN  
749 KBDDOSSETCP  
750 KBDDOSLOADINSTANCE  
751 KBDDOSSYNCH  
752 KBDDOSREGISTER  
753 KBDDOSSTRINGIN  
754 KBDDOSGETSTATUS  
755 KBDDOSSETSTATUS  
756 KBDDOSGETFOCUS  
757 KBDDOSFLUSHBUFFER  
758 KBDDOSXLATE  
759 KBDDOSSWITCHFGND  
760 KBDDOSSHELLINIT  
761 KBDDOSCLOSE  
762 KBDDOSFREEFOCUS  
763 KBDDOSFREE  
764 KBDDOSDEREGISTER  
765 KBDDOSSETFGND  
766 KBDDOSPEEK  
767 KBDDOSOPEN  
768 KBDDOSGETHWID

769 KBDDOSSETHWID  
770 VIODOSENDPOPUP  
771 VIODOSGETPHYSBUF  
772 VIODOSGETANSI  
773 VIODOSFREE  
774 VIODOSSETANSI  
775 VIODOSDEREGISTER  
776 VIODOSSCROLLUP  
777 VIODOSPRTSC  
778 VIODOSGETCURPOS  
779 VIODOSWRTCELLSTR  
780 VIODOSPOPUP  
781 VIODOSSCROLLRT  
782 VIODOSWRTCHARSTR  
783 VIODOSAVS\_PRTSC  
784 VIODOSSETCURPOS  
785 VIODOSSRFUNBLOCK  
786 VIODOSSRFBLOCK  
787 VIODOSSCRUNLOCK  
788 VIODOSWRTTTY  
789 VIODOSSAVE  
790 VIODOSGETMODE  
791 VIODOSSETMODE  
792 VIODOSSCRLOCK  
793 VIODOSREADCELLSTR  
794 VIODOSSAVREDRAWWAIT  
795 VIODOSWRTNATTR  
796 VIODOSGETCURTYPE  
797 VIODOSSAVREDRAWUNDO  
798 VIODOSGETFONT  
799 VIODOSREADCHARSTR  
800 VIODOSGETBUF  
801 VIODOSSETCURTYPE  
802 VIODOSSETFONT  
803 VIODOSHETINIT  
804 VIODOSMODEUNDO  
805 VIODOSSSWSWITCH  
806 VIODOSMODEWAIT  
807 VIODOSAVS\_PRTSCTOGGLE  
808 VIODOSGETCP

809 VIODOSRESTORE  
810 VIODOSSETCP  
811 VIODOSSHOWBUF  
812 VIODOSSCROLLLF  
813 VIODOSREGISTER  
814 VIODOSGETCONFIG  
815 VIODOSSCROLLDN  
816 VIODOSWRTCHARSTRATT  
817 VIODOSGETSTATE  
818 VIODOSPRTSCTOGGLE  
819 VIODOSSETSTATE  
820 VIODOSWRTNCELL  
821 VIODOSWRTNCHAR  
822 VIODOSSHELLINIT  
823 VIODOSASSOCIATE  
824 VIODOSCREATEPS  
825 VIODOSDELETESETID  
826 VIODOSGETDEVICECELLSIZE  
827 VIODOSGETORG  
828 VIODOSCREATELOGFONT  
829 VIODOSDESTROYPS  
830 VIODOSQUERYSETIDS  
831 VIODOSSETORG  
832 VIODOSQUERYFONTS  
833 VIODOSSETDEVICECELLSIZE  
834 VIODOSSHOWPS  
835 VIODOSGETPSADDRESS  
836 VIODOSQUERYCONSOLE  
837 VIODOSREDRAWSIZE  
838 VIODOSGLOBALREG  
839 XVIODOSSETCASTATE  
840 XVIODOSCHECKCHARTYPE  
841 XVIODOSDESTROYCA  
842 XVIODOSCREATECA  
843 VIODOSHECKCHARTYPE  
844 XVIODOSGETCASTATE  
845 BVSDOSMAIN  
846 BVSDOSREDRAWSIZE  
847 BVSDOSGETPTRDRAWNAME



848 ANSIDOSINJECT  
849 ANSIDOSKEYDEF  
850 ANSIDOSINTERP  
851 BKSDOSMAIN  
852 BMSDOSMAIN  
853 MOUDOSGETHOTKEY  
854 MOUDOSSETHOTKEY  
855 SMGDOSSMSYSINIT  
856 SMGQHKEYBDHANDLE  
857 SMGQHMOUSEHANDLE  
858 CHRQueueRamSem  
859 CHRArray  
860 CHRPIDArray  
861 CHRInitialized  
862 CHRArraySize  
863 CHRBVSGLOBAL  
864 CHRSMGINSTANCE  
865 CHRBVHINSTANCE  
115 THK32ALLOCMEM  
116 THK32FREEMEM  
117 THK32ALLOCSTACK  
262 THK32FREESTACK  
546 THK32STRLEN  
547 THK32\_UNITHUNK  
578 HT16\_CLEANUP  
579 HT32\_STARTUP  
581 HT32\_CLEANUP  
590 DOS32PMPOSTEVENTSEM  
591 DOS32PMWAITEVENTSEM  
593 DOS32PMREQUESTMUTEXSEM  
595 DOS32PMWAITMUXWAITSEM  
596 DOS32PM16SEMCHK  
866 THK32ALIASMEM  
867 THK32FREEALIAS  
868 THK32ALLOCVARLEN  
869 THK32HANDLEBOUNDARY  
870 THK32HANDLESTRING  
871 THK32DEALLOC  
872 THK32XHNDLR  
873 DOS32SETTEXTLIBPATH

874 DOS32QUERYEXTLIBPATH  
875 DOS32PM16SEMRST  
876 DOS32SYSCTL  
998 DOSSETEXTLIBPATH  
999 DOSQUERYEXTLIBPATH  
1000 T32EXITLIST  
1001 T32ALLOCPROTECTEDMEM  
1002 T32ALIASMEM  
1003 T32ALLOCMEM  
1004 T32ALLOCSHAREDMEM  
1005 T32GETNAMEDSHAREDMEM  
1006 T32GETSHAREDMEM  
1007 T32GIVESHAREDMEM  
1008 T32FREEMEM  
1009 T32SETMEM  
1010 T32QUERYMEM  
1011 T32QUERYMEMSTATE  
1012 T32OPENVDD  
1013 T32REQUESTVDD  
1014 T32CLOSEVDD  
1015 T32CREATETHREAD  
1016 T32DYNAMICTRACE  
1017 T32DEBUG  
1018 T32QUERYPROCADDR  
1019 T32CREATEEVENTSEM  
1020 T32OPENEVENTSEM  
1021 T32CLOSEEVENTSEM  
1022 T32RESETEVENTSEM  
1023 T32POSTEVENTSEM  
1024 T32WAITEVENTSEM  
1025 T32QUERYEVENTSEM  
1026 T32CREATEMUTEXSEM  
1027 T32OPENMUTEXSEM  
1028 T32CLOSEMUTEXSEM  
1029 T32REQUESTMUTEXSEM  
1030 T32RELEASEMUTEXSEM  
1031 T32QUERYMUTEXSEM  
1032 T32CREATEMUXWAITSEM  
1033 T32OPENMUXWAITSEM

1034 T32CLOSEMUXWAITSEM  
1035 T32WAITMUXWAITSEM  
1036 T32ADDMUXWAITSEM  
1037 T32DELETEMUXWAITSEM  
1038 T32QUERYMUXWAITSEM  
1039 T32QUERYSYSINFO  
1040 T32WAITTHREAD  
1041 T32GETRESOURCE  
1042 T32FREERESOURCE  
1043 T32EXCEPTIONCALLBACK  
1044 T32QUERYPAGEUSAGE  
1045 T32FORCESYSTEMDUMP  
1046 TI32ASYNCTIMER  
1047 TI32STARTTIMER  
1048 T32QUERYABIOSSUPPORT  
1049 T32QUERYMODFROMEIP  
1050 T32ALIASPERFCTRS  
1051 T32CONFIGUREPERF  
1052 T32DECONPERF  
1053 T32REGISTERPERFCTRS  
1054 T32QUERYSYSSTATE  
1055 T32IREAD  
1056 T32IWRITE  
1057 T32TMRQUERYFREQ  
1058 T32TMRQUERYTIME  
1059 T32IMONREAD  
1060 T32IMONWRITE  
1061 T32QUERYRESOURCESIZE  
1062 T32PROFILE  
1063 T32SETSIGNALEXCEPTIONFOC  
1064 T32SENDSIGNALEXCEPTION  
1065 T32STARTTIMER  
1066 T32STOPTIMER  
1067 T32ASYNCTIMER  
1068 T32INITIALIZEPORTHOLE  
1069 T32QUERYHEADERINFO  
1070 T32QUERYPROCTYPE  
1071 T32IEXITMUSTCOMPLETE  
1072 T32ICACHEMODULE  
1073 T32DLLTERM

1074 T32IRAISEEXCEPTION  
1075 T32ACKNOWLEDGESIGNALEXC  
1076 T32QUERYDOSPROPERTY  
1077 T32SETDOSPROPERTY  
1078 T32SETFILELOCKS  
1079 T32CANCELLOCKREQUEST  
1080 T32KILLTHREAD  
1081 TQUERYRASINFO  
1082 T32DUMPPROCESS  
1083 T32SUPPRESSPOPUPS  
1084 T32IPROTECTWRITE  
1085 T32PROTECTSETFILELOCKS  
1086 T32IPROTECTREAD  
1087 T32PMPOSTEVENTSEM  
1088 T32PMWAITEVENTSEM  
1089 T32PMREQUESTMUTEXSEM  
1090 T32PMWAITMUXWAITSEM  
1091 T32PM16SEMCHK  
1092 T32ALLOCTHREADLOCALMEMORY  
1093 T32FREETHREADLOCALMEMORY  
1094 T32SETEXTLIBPATH  
1095 T32QUERYEXTLIBPATH  
1096 T32PM16SEMRST  
1097 T32VERIFYPIDTID  
1098 T32SYSCTL

---

## OS/2 Fix Pack to Build Level Cross-reference

The following tables cross-references some of the public fix packs and GA versions of OS/2 with their internal kernel build level.

### Notes:

Some fix packs use the same kernel build level when updates are confined to modules other than OS2KRNL.

The build level for a system may be determined using the command:

```
VER /R
```

The build level of a system module may be determined using the command:

```
BLDLEVEL <file name>
```

The fix pack level for a system may be determined using the **SYSLEVEL** command. The following example shows the system level information for OS/2 Warp V4.0 fix pack 6:

```
C:\OS2\INSTALL\SYSLEVEL.FPK
                                OS/2 Warp 4 Service Level
Version 1.00      Component ID 566933010
Type Fixpak
Current CSD level: XR0M006
Prior   CSD level: XR0M006
```

[OS/2 V2.11 Build Level Cross-reference](#)

[OS/2 Warp V3.0 Build Level Cross-reference](#)

[OS/2 Warp V4.0 Build Level Cross-reference](#)

---

## OS/2 V2.11 Build Level Cross-reference

<i>Version</i>	<i>Build Level</i>
2.11 GA	6.617
XR_A076	6.653
XR_A080	6.653
XR_A090	6.656
XR_A092	6.658
XR_A095	6.661
XR_A096	6.660
XR_B097	6.664
XR_B098	6.665
XR_B099	6.667
XR_B100	6.668
XR_B101	6.669
XR_B102	6.670
XR_B103	6.671
XR_B104	6.672
XR_B105	6.673
XR_B106	6.674
XR_B107	6.675
XR_B108	6.676
XR_B109	6.677

---

## OS/2 Warp V3.0 Build Level Cross-reference

<i>Version</i>	<i>Build Level</i>
Warp GA	8.162
Warp Full Pack	8.200
Warp Connect	8.209
XR_W005	8.213B
XR_W007	8.230
XR_W008	8.230
XR_W009	8.234
XR_W010	8.234
XR_W011	8.235
XR_W012	8.236
XR_W013	8.237
XR_W014	8.238
XR_W016	8.240
XR_W017	8.241
XR_W018	8.242
XR_W019	8.243
XR_W020	8.244
XR_W021	8.245
XR_W022	8.246
XR_W023	8.247
XR_W024	8.248
XR_W025	8.249
XR_W026	8.250
XR_W027	8.251
XR_W028	8.252
XR_W029	8.253
XR_W032U	8.256_uni
XR_W033U	8.257_uni
XR_W034U	8.258_uni
XR_W035U	8.259_uni
XR_W036U	8.260_uni

---

# OS/2 Warp V4.0 Build Level Cross-reference

<i>Version</i>	<i>Build Level</i>
4.0 GA	9.023
XR_M000	9.024
XR_M001	9.025
XR_M002	9.026
XR_M003	9.027
XR_M004	9.028
XR_M005	9.029
XR_M006	9.030

---

## Glossary

The following terms and acronyms have been referenced in this document.

- AAB
- Absolute symbol
- Block Id
- BMP
- BPB
- Breakpoint
- cbargs
- CBIOS
- CDA
- CDIB
- CDS
- CRI
- Compatibility Region Mapping Algorithm (CRMA)
- Command Subtree (Csid)
- Context
- DEM
- DosHlp
- DPB
- FAT
- FSC
- Gate
- GDT
- Global Info Segment
- hal
- har
- hco
- hmte
- hob
- hptda
- HWND
- IDT
- Interrupt Vector
- IPE
- JFN

JFN Table  
KSEM  
LDT  
Local Info Segment  
Maps  
MFT  
MQ  
MTE  
OTE  
PAI  
paragraph  
PDB  
PF  
PIB  
Pid  
Process  
Pseudo-Object  
PSP  
PTDA  
PTE  
PTree  
PWND  
QMSG  
RIP  
RAS  
RLR  
RMP  
SAS  
SFN  
SFT  
SMS  
STDA  
Scheduler  
Session  
Screen Group  
SGCB  
Slot  
SMTE  
SQMSG  
STE  
Symbols  
Symbol Groups  
Task  
TCB  
Thrashing  
Thread  
Thunk  
TIB  
Tid  
TLB  
TLMA  
Tracepoint  
TSD  
TSS  
UVIRT  
VMAH  
VMAL  
VMAR  
VMCO  
VMKH  
VMOB  
VDM  
VP  
VPB  
WND  
Zombie

-----

(No title)



The term loader applies to two distinct components under OS/2:

OS2LDR	This is the OS2KRNL loader. One of its functions is to load the OS2KRNL module at boot time. After the system has booted OS2LDR provides the <a href="#">CBIOS</a> layer for the kernel.
System Loader	This is a component of the kernel. It is responsible for loading program modules, DLLs, Device Drivers and File System Drivers.

The logging facility discussed in this section applies to the System Loader.

---

## (No title)

Throughout this chapter the term **debugger** is used loosely to mean any of the following where ambiguity is not a problem:

- Debug Kernel (HSTRICT or ALLSTRICT).
  - The debugger component within the debug kernel.
  - The debugging console.
- 

## (No title)

A simple numeric expression is one that resolves to a single integer value, for example:

-4  
55c7

Compare this with an address expression that has in addition an address operator (&, %, %, #) and possibly involves more than one integer value, for example:

&1fc:45  
#1f:445  
%30045  
%%15c

---

## (No title)

Feature 82818 introduces the Kernel Debugger [.MK command](#). 82818 is supplied as an APAR fix to:

OS/2 Warp V3.0:	as PJ18364 in fix pack 7
OS/2 V2.11:	as PJ16805 in fix pack 90

---

## (No title)

The control block formatting conventions have been chosen to aid the user of the Kernel Debugger and Dump Formatter.

Each control block is presented in tabular form with 5 columns used as follows:

<i>name</i>	Field name, usually taken from the C header or MASM include file definition.		
<i>Off</i>	Offset from the beginning of the structure. The offset is of the form <i>x.y</i> where <i>x</i> is the signed hexadecimal byte offset from the beginning of the structure and <i>y</i> is the bit offset from the high-order bit of the byte.		
<i>Leng</i>	Hexadecimal length of the field.		
<i>Type</i>	The field type, for the purposes of displaying storage using the <a href="#">D command</a> . The following values are used:		
	S	Complex structure. Choose display command to best suit your needs.	
	D	Double word. Use <b>DD</b> to format the field correctly.	
	W	Word. Use <b>DW</b> to format the field correctly.	
	B	Byte. Use <b>DB</b> to format the field correctly.	
	A	ASCII byte string. Use <b>DA</b> to format the field correctly.	
<i>blank</i>	A blank value appears when a field does not begin or end on a byte boundary. In this case format the field from the previous field for which a type value is given. Such bit fields are presented in an order assuming this instruction is followed. Attempts to display bit fields in other ways may lead to a great deal of confusion!		
<i>Description</i>	field description taken usually from the header or include file.		

A null row is used to indicate an overlay definition of the same control block.

Flag fields are separately formatted in tabular form.

Where a flag field represents a bit mask, the mask is given in hexadecimal and is assumed to indicated that corresponding bits are set to be in effect. Exceptions are specifically notes in the description.

When the flag field take numerical values then they will be shown in either hexadecimal (prefixed with **0x**) or decimal depending on the C or MASM definitions.

## (No title)

A PM **Send Message Structure (SMS)** is used by **WinSendMessage** to enqueue a synchronously sent message. SMSs are chained from the [MQ](#) of both the sender and receiver.

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

## (No title)

A PM **Message Queue Header (MQ)** is used as an anchor for message processing for a given PM Application's message thread. The MQ is created when a threads calls **WinCreateMsgQueue**.

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

## (No title)

A PM **Queue Message (QMSG)** is used by **WinPostMsg** to enqueue an asynchronously sent message to a thread's message queue. QMSGs are chained from the [MQ](#) of the receiver in a circular array.

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

---

## (No title)

A PM **System Queue Message (SQMSG)** is used by the **PMDD.SYS** device driver to enqueue messages, which represent system input activity, to the system input queue.

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

---

## (No title)

A **hwnd** is a 32-bit pointer to a [WND](#) structure.

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

---

## (No title)

An **hwnd** is the handle to a [WND](#) structure. This is returned to an application when it uses **WinCreateWindow** and is used for subsequent PM API calls that affect the window.

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

---

## (No title)

A PM **Application Anchor Block** is allocated in the [Thread Local Memory Area](#) when a PM application thread creates a message queue. The AAB contains a pointer to the [MQ](#) which allows PM to find the MQ in any context. This is particularly useful to the debugger since it also allows the MQ of any PM thread in the system since the TLMA is saved in a thread's [TCB](#).

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

---

## (No title)

The **Thread Local Memory Area (TLMA)** is an area of private arena memory that is instantiated at a thread level. This is achieved by copying the contents of the TLMA to dfff:0024 when a thread switch occurs. The TLMA contents are saved in the [TCB](#) at **TCBTLMA**.

Storage is allocated from the TLMA by using the **DosAllocThreadLocalMemory** API.

See [Context Switching](#) for more information.

---

## (No title)

A PM **Window Structure (WND)** is used by PM to represent a window. When an application uses **WinCreateWindow** the WND is created and the [hwnd](#) is returned to the user. The WND contains information about a window's hierarchy, and associated [MQ](#).

See [Exploring 32-bit Presentation Manager Under WARP](#) for more information.

---

## (No title)

A **tracepoint** is designated location in system or application code where the [System Trace Facility](#) will gather data for logging by the [STDA](#).

Tracepoints may be implemented statically by use of the [DosSysTrace](#) API or dynamically through use of the [Dynamic Trace Customiser](#).

System defined tracepoints are documented in the [System Tracepoints Reference](#).

---

## (No title)

The term **Zombie** is used to describe a terminal condition of a thread or process. There is a strict operating system definition and two colloquial uses:

- The strict system definition refers to a process that has terminated but whose [PTDA](#) has been retained on the zombie queue ([\\_pPTDAFirstZombie](#)) because the process status byte (LISEG+0xa) indicates that its parent wishes to collect termination information through **DosWaitChild**. The dead child is retained on the zombie queue until either the parent dies or issues **DosWaitChild**.
  - Zombie is also commonly used to refer to a terminating thread or process that has blocked after the application has returned to the operating system. Usually this implies a problem freeing memory because one or more pages have been long-term locked by a device driver.
  - The third use of zombie refers to any process that is anonymous. Internal thread, VDMs, and terminating threads can be anonymous.
- 

## (No title)

The Compatibility BIOS (CBIOS) is a layer of code in the OS2LDR that presents a hardware independent interface to the BIOS for the OS2KRNL. The interface to the OS2KRNL is provided through a set of entry points called [Dos Helper Functions](#).

---

## (No title)

A BIOS Parameter Block is used for low level Disk I/O calls to the BIOS.

For further information see:

The [.D BPB Command](#) in the Kernel Debugger and Dump Formatter Command Reference.

The [BPB Structure](#) in the System Reference.

---

## (No title)

Kernel Semaphores are a form of semaphore, similar to the application 32-bit semaphore, used by kernel routines for longer term blocking. Kernel Semaphores provide addition funtion over the simple blocking mechanism, which includes:

- Priority inversion protection.

- Ownership auditability.

For additional information see the following:

- The [.D KSEM](#) command.

- The [.PB](#) command.

- The [KSEM strutures](#) in the **System Reference**.

---

## (No title)

Fixed Allocation Table (FAT) file system is the default filing system supported by OS/2. Support for **FAT** is always present, regardless of any installed file systems.

---

## (No title)

Internal Processing Errors (IPEs) are unrecoveralble error conditions detected by the system while running in ring 0. The may arise from inconsistencies detected by the OS/2 Kernel or from traps occuring in any ring 0 code (Kernel, Installable File System Drivers and Device Drivers).

When the system detects an **IPE** it enters a routine called **panic** where an error message is formatted and displayed and the system is halted.

---

## (No title)

Reliability, Availability and Serviceability (RAS) is an acronym that refers to diagnostic and service support within OS/2. Frequently it is used as a synonym for the adjective *diagnostic*.

---

## (No title)

A **task** is hardware architected thread of execution. The INTEL architecture allows for multiple independent tasks to co-exit and provides the [task gate](#) mechanism as a means of switching between tasks. Tasks are represented to the hardware by the [TSS](#).

The characteristics of a **task** are very similar to that of the OS/2 [process](#). Protect-mode processes however, tend to run under a single task in OS/2 and implement switching through the more efficient software managed [context](#) switching mechanism.

Only [VDMs](#) and error recovery processes run as independent **tasks**.

See the **INTEL486 Programmer's Reference** for more information.

(No title)

A **gate** descriptor is one that defines to the hardware a means of entering code that executes at a more privileged level of authority. Four types of **gate** are defined:

<b>Call Gate</b>	The subject of a <b>CALL</b> instruction. Typically used to implement operating system and device driver application programming interfaces (APIs). Device drivers may create <i>Call gates</i> dynamically using the <b>DosDynamicAPI</b> facility.
<b>Task Gate</b>	The subject of a call or exception where a (hardware assisted) <b>task</b> switch is required.
<b>Interrupt Gate</b>	The subject of a hardware or software generated interrupt. Typically an <b>Interrupt gate</b> will switch execution to an interrupt handler when a device presents an interrupt.
<b>Trap Gate</b>	The subject of a trap exception. Used to handle programming errors.

(No title)

A **Virtual Dos Machine (VDM)** is a type of process that runs in an emulated DOS environment using the **DEM** component of OS/2.

(No title)

The **Thread Information Block (TIB)** is a supplemental thread related control block made accessible to ring 3 programs. It contains thread information obtained from the thread's **TCB** and acts as an anchor for exception-handlers registered for the thread.

The **TIB** may be located from **TCBptib (TCB + 0x10)** using the Dump Formatter or Kernel Debugger.

A program gains access to the **TIB** along with the **PIB** by calling the **DosGetInfoBlocks** API.

(No title)

The **Process Information Block (PIB)** is a supplemental process related control block made accessible to ring 3 programs. It contains process status information obtained from the process' **PTDA**.

The **PIB** may be located from **ptda\_avatib (PTDA + 0x28)** using the Dump Formatter or Kernel Debugger

A program gains access to the **PIB** along with the **TIB** by calling the **DosGetInfoBlocks** API.

(No title)

The **Task State Segment (TSS)** is a hardware architected control block that is used for two purposes:

1. To implement the privileged level transition mechanism initiated with a **Call Gate** instruction.
2. To provide a register save area for hardware task switching initiated with a call to a **Task Gate**.

In general OS/2 does not use the hardware task switching mechanism, so **TSSs** are few. It does however use the **TSS** for implementing Application Programming Interfaces (APIs) in the system.

A **TSS** may be formatted using the Kernel Debugger and Dump Formatter [DT command](#).

-----

## (No title)

The **System Trace Data Area (STDA)** is a circular buffer used to record trace events. The STDA may be located from the [SAS](#).

-----

## (No title)

The Physical Arena Information structure (PAI) describes ranges of physical memory to memory management.

Physical memory ranges mapped by RAM is described by the [PAI](#) pointed to by the [SAS](#).

There are two **PAIs** located at the following symbols:

### **pgPageablePAI.**

This described ranges of physical memory available for pageable memory allocations. Normally two ranges are described:

Below 1Mb

16Mb and above

### **pgResidentPAI.**

This described ranges of physical memory available for backing resident memory allocations. Normally three ranges are described:

Below 1Mb

Below 16Mb

16Mb and above

-----

## (No title)

The Interrupt descriptor table (IDT) is a hardware architected structure that comprises a table of [gate descriptors](#), one for each [interrupt vector](#). The low numbered entries are defined by the hardware architecture and dedicated to exception management.

Under OS/2 one IDT is allocated for the entire system except for VDMs in which multiple IDTs per VDM are possible. A VDM will allocate one IDT per DPMI client, but if it does not use DPMI then the common system IDT is used. When multiple IDTs are used in a VDM then only the entries not reserved for H/W exceptions and interrupts are allowed to differ.

The Kernel Debugger's [V command](#) may be used to intercept system exception handlers.

-----

## (No title)

An **interrupt vector** is presented to the processor when an interrupt is generated either externally by the Programmed Interrupt Controller or internally within the processor chip itself. It is used by the processor as an index into the [IDT](#) to determine which interrupt routine should be dispatched.

The processor reserves **vectors** 0 - 31 to correspond to hardware architected exceptions 0 through 31. **Vectors** 32 - 255 are reserved for I/O interrupts, which are presented to the processor by the Programmed Interrupt Controller when the one of its **IRQ** lines is triggered. The correspondence between **vectors** and **IRQs** is defined during system initialisation as follows:

IRQs 0 - 7                      vectors 0x50 - 0x57

IRQs 8 - 15                    vectors 0x70 - 0x77

Thus a keyboard interrupt, which is assigned to IRQ 1 under the IBM PC architecture will be handled by the interrupted handler whose [interrupt gate](#) is assigned to **IDT** descriptor **0x51**.

See the Dump Formatter and Kernel Debugger [DI command](#) for information on displaying **IDT** entries.

-----

## (No title)

The Local Information Segment is a per-process control block that records the current status of the process. It is imbedded in the [PTDA](#) and is also mapped by [LDT](#) descriptor **0xdfff**.

-----

## (No title)

The **Global Information Segment (GISEG)** is a single instance control block that records the current session status, date and time, trace status and version of the system.

The system maintains two copies of the Global Information Segment to fence against system damage.

The selector for the **GISEG** may be located from the [SAS](#). See the Dump Formatter and Kernel Debugger [.A command](#).

The **GISEG** is also mapped locally per-process by [LDT](#) descriptor **0xdfff4**.

-----

## (No title)

A **Register Information Packet (RIP)** is an entity used to describe the size and offset of a register in a system stack frame. **RIPs** are located in a [CRI](#).

-----

## (No title)

The **Client Register Information (CRI)** is a table of **Register Information Packets (RIPs)** that describe the offset and length of each register that is stored in a ring 0 stack frame on entry to the kernel. This level of indirection allows kernel routines to access entry registers regardless of the stack frame type, of which there are a number, for example:

System Entry Frames from API calls

Trap Frames from traps and exceptions

Interrupt Frames from the interrupt manager

VDM Stack Frames



## Kernel Stack Frames

Each **TCB** points to a **CRI** and the associated stack frame from **TCB\_pcriFrameType** (**TCB** + 0x38) and **TCB\_pFrameBase** (**TCB** + 0x3c) respectively.

## (No title)

**Context** (or thread context) refers the *view* of the system any given **thread** has. Only one thread context may be current at any time.

*Switching contexts* refers to the process of preparing the system for another thread to run. From an application program's perspective this implies restoring its registers and ring 2 and 3 stacks when it is given the opportunity to run again. From the system's perspective, restoration of an application's registers and stacks is done after the context switch, by the dispatcher, on exiting kernel mode. Not every context switch is followed by exiting kernel mode. For example, if another thread in the same process is in critical section (but blocked) then the new thread enters **crt** state and the scheduler is called to select yet another thread.

Context switching includes the following system actions:

- Updating GDT descriptor entries 28, 30, 38 and 150b, which point to
  - the current process' **LDT**,
  - the current threads ring 0 stack,
  - the current thread's floating point emulator work area,
  - the current thread's **TIB**. (By default the **FS** selector is loaded with 150b).

**Note:** The **LDT** selector is only updated when the process changes with a context switch, that is, for a process context switch.

- Updating page directory and tables for a process context switch.
- Updating the **TR** register if the process switch involves a task switch (normally only **VDMs**).
- Updating the current **TSS** ring 0 and ring 2 stack addresses.
- Updating system copies of the Global and Local Information Segments.
- Copying the Local Information Segment from the incoming PTDA and the Thread Local Memory Area from the incoming TCB to the segment mapped by LDT selector **dfff**.

### Note:

Besides addressing the current ring 0 stack, selector 30 also addresses the current thread's scheduling control blocks. In particular: the **PTDA**, **TCB** and **TSD**. This is done by aliasing selected address ranges from selector 30 to those of the true PTDA, TCB and TSD in the system arena global memory for the current context. The system defines a dummy module containing a hard-coded PTDA. The symbols of this module have the same name as those of the fields in the PTDA. The system arranges for this to map the PTDA addressed by selector 30. This trick allows the system to refer to PTDA fields for the current context without regard for which process is current, simply by using the field names as public symbols. The user may use the same symbols for referencing the PTDA but these are only valid for the current system context. To access PTDA fields in other contexts the following technique can be used:

```
-----
Note that the current PTDA is located at PTDA_Start
##.p *
  Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
*0025  0004  0002  0004  0001  blk  0300  7b7c8000  7bbc4080  7bbe8a90  1fc4  16  someprog
The current thread slot is 25
We wish to know the thread that has entered critical section in process of
thread slot 40. The address of the critical section TCB is
saved in ptda_ptCBCritSec and the thread ordinal and slot number
are the first two words of the TCB.
##.p 40
  Slot  Pid  Ppid  Csid  Ord  Sta  Pri  pTSD      pPTDA      pTCB      Disp  SG  Name
  0040  0012  0002  0012  0001  blk  0500  7b7d6000  7b9e4020  7b9c8a70  1eb8  10  userprog
##dw  %(DW(%7b9e4020+ptda_ptcbcritsec-ptda_start)) 12
%7b9c8de0 0002 0041
Thread 2 of 12 or thread slot 41 is in critical section
```

```
##.p 41
Slot  Pid  Ppid Csid Ord  Sta Pri  pTSD      pPTDA      pTCB      Disp SG Name
0041  0012 0002 0012 0002 blk 0800 7b7da000 7b9e4020 7b9c8de0 1ed4 10 userprog
-----
```

Refer to the Kernel Debugger and Dump Formatter [.P](#) and [.S](#) commands for more information.

## (No title)

**Doshlp** services comprise a set of hardware dependent service routines established during system initialisation for use by the **OS2KRNL** and user programs via the **OEMHLP\$** device driver. Many of the **Doshlp** services deal with device dependent BIOS behaviour and therefore provide a device independent interface to the BIOS.

## (No title)

A **breakpoint** is a location in a program where execution is suspended and control is given to a debugging tool.

The **INTEL** architecture supports two implementations of breakpoints for debugging purposes:

The software generated breakpoint using the **INT 3** instruction;

The hardware generated breakpoint using the **Debugging Registers**.

The use of software breakpoints require code modification, whereas the use of debugging registers does not. However, the number of predefined software breakpoints is potentially unlimited whereas there are only 4 breakpoints specifiable using **Debugging Registers**.

A further distinction between the two types is that software breakpoints only intercept the execution of a particular instruction path, whereas **Debugging Registers** may be used, in addition, to intercept data fetches and stores from a particular location in virtual memory.

The Kernel Debugger supports both implementations of breakpoints through the use of the:

[BR command](#), which uses **Debugging Registers**, and

[BP command](#), which uses **INT 3** instructions.

The Kernel Debugger limits the predefinition of **BP** breakpoints to **10**, however the programmer may code as many additional **INT 3** instructions into his/her program as desired.

The Kernel Debugger refers to breakpoints explicitly set by the **BP** and **BR** commands as *sticky* (implying a certain permanence about them). The [G command](#) may have one or more temporary breakpoints established when one or more *stop addresses* are specified. These are referred to as *go* breakpoints. Once the Kernel Debugger breaks in *go* breakpoints are removed. The internal operation of the Kernel Debugger may also necessitate the use of the occasional *temporary* breakpoints when instruction tracing (see the [T](#) and [P](#) commands). These are set implicitly and discarded without the user being aware of their existence. *Go* and *temporary* breakpoints are created using the **INT 3** instruction. *Go* and *sticky BP* breakpoints count towards the Kernel Debugger imposed limit of **10**, but *temporary* breakpoints only ever exist singly so do not.

## (No title)

A symbol **map** is created from symbolic name information generated by a program compiler and converted for use by the Dump Formatter and Kernel Debugger by the linkage editor and **MAPSYM** utilities. This allows program code and data locations to be referred to by name as well as by address.

## (No title)

A symbol group is the set of [symbols](#) that are defined within a program segment. Frequently a program segment is given its own selector at load time.

---

## (No title)

An **symbol** is the name given to a program code or data location that has been made public by the programmer. Such symbolic definitions appear in the map file output from the linkage editor. They may be referenced in the Dump Formatter and Kernel Debugger using the [L command](#) when the map file is converted to a symbol file using the **MAPSYM** utility.

---

## (No title)

An **Absolute symbol** is a symbolised constant value that has been made public by the programmer. Such symbolic definitions appear in the map file output from the linkage editor and may be referenced in the Dump Formatter and Kernel Debugger using the [LA command](#) when the map file is converted to a symbol file using the **MAPSYM** utility.

---

## (No title)

A **System File Number (SFN)** is the system-wide unique handle by which an open file system object is known. It is the offset into the [SFT](#) segment that locates the corresponding **SFT** entry.

Refer to the following for more related information

[.A Kernel Debugger and Dump Formatter command.](#)

[.D Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

A Job File Number (**JFN**) is a handle for open file system objects, unique within the process that opened the file system object. The **JFN** is returned by **DosOpen**. It is used and an index into the [JFN Table](#) to locate the corresponding [SFT](#) handle.

---

## (No title)

A **Patricia Tree (PTREE)** is a form of tree structure designed to offer a fast look-up facility for generically specified keys. In OS/2 a modified form of the **PTREE** is use to manage [MFTs](#) for fast path-name look-up.

---

## (No title)

The **System Anchor Segment (SAS)** is a central system control block use to anchor control blocks for major system components such as:

- File systems
- Device Drivers
- Scheduler
- Memory management

The **SAS** is built at the beginning of the segment addressable from selector **70** and **78**.

Refer to the following for more detailed information

[.A Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

A **Master File Table (MFT)** entry is used to associate path names with open files (**SFTs**) and lock records (**RLRs**). The **MFTs** are managed in a **PTREE** structure, which is locatable from the **SAS**.

See also related structures:

- [CDS](#)
- [DPB](#)
- [SFT](#)
- [FSC](#)
- [VPB](#)

Refer to the following for more detailed information

- [.A Kernel Debugger and Dump Formatter command.](#)
- [.D Kernel Debugger and Dump Formatter command.](#)
- MFT** control block format

---

## (No title)

A **Record Lock Record (RLR)** describes a locked region of a file system record. **RLRs** are chained from the related and point to the associated **SFT**. They record the owner of the record lock.

See also related structures:

- [CDS](#)
- [DPB](#)
- [SFT](#)
- [FSC](#)
- [VPB](#)

Refer to the following for more detailed information

[RLR](#) control block format

-----

## (No title)

A **System File Table (SFT)** entry is used to describe the attributes of each instance of an open file system object. **SFTs** are stored in a segment directly locatable from the [SAS](#). **SFTs** are indirectly locatable from the [JFN Table](#) imbedded in the [PTDA](#) of each process that opens a file system object.

See also related structures:

- [CDS](#)
- [DPB](#)
- [MFT](#)
- [FSC](#)
- [VPB](#)

Refer to the following for more detailed information

- [.A Kernel Debugger and Dump Formatter command.](#)
- [.D Kernel Debugger and Dump Formatter command.](#)
- [SFT](#) control block format

-----

## (No title)

A **Job File Number Table (JFT)** entry is assigned to each open file system object within a process. The **JFT** provides a cross-reference to the handle for the corresponding [SFT](#). The **JFT** is locatable from the [PTDA](#)field **JFN\_pTable** (**PTDA** +0x5b8 (H/R: +0x5b0)) for each process.

The **JFT** is initially allocated within the **PTDA** at label **JFN\_Table** (**PTDA** +0x35e) with 20 entries. If this is expanded by use of the **DosSetMaxFH** then **JFN\_pTable** is updated to point to the new table.

See also related structures:

- [CDS](#)
- [DPB](#)
- [MFT](#)
- [FSC](#)
- [VPB](#)

-----

## (No title)

A **Volume Parameter Block (VPB)** is used to store volume information associated with a file system object. All **VPBs** are contained within a

single segment locatable from the [SAS](#). Most file system structures contain a **VPB** handle for an associated volume. The handle is used as an offset into the **VPB** segment.

See also related structures:

- [CDS](#)
- [MFT](#)
- [SFT](#)
- [DBP](#)
- [FSC](#)

Refer to the following for more detailed information

- [.A Kernel Debugger and Dump Formatter command.](#)
- [.D Kernel Debugger and Dump Formatter command.](#)
- VPB** control block format

-----

## (No title)

A **Current Directory Structure (CDS)** is used to store file system information about the current directory per drive of each process.

Each **CDS** is managed in an [RMP](#) segment. The [PTDA](#) for each process contains an imbedded array of 26 **CDS** handles, one for each drive. The **CDS RMP** segment may be located from the [SAS](#).

See also related structures:

- [MFT](#)
- [SFT](#)
- [DPB](#)
- [FSC](#)
- [VPB](#)

Refer to the following for more detailed information

- [.A Kernel Debugger and Dump Formatter command.](#)
- [.D Kernel Debugger and Dump Formatter command.](#)
- CDS** control block format

-----

## (No title)

The **Codepage Data Information Block (CDIB)** contains country-specific constant information relating to screen, keyboard and printer devices. The **CDIB** is built from information derived directly from **CONFIG.SYS** statements.

The **CDIB** may be located from the [SAS](#).

-----

## (No title)

A **File System Control Block (FSC)** represents an installed file system (IFS). The FSC contains a table of entry points implemented by the file system driver ([FSD](#)). All FSCs are located in a single segment whose selector may be obtained from the the [SAS](#). see [.A command](#)

See also related structures:

- [CDS](#)
- [MFT](#)
- [SFT](#)
- [DPB](#)
- [VPB](#)

Refer to the following for more detailed information

[FSC](#) control block format

## (No title)

A File System Driver (FDS) is a special load module that implements an installed file system (IFS). **FSDs** are loaded during system initialisation when [IFS=](#) statement of **CONFIG.SYS** is encountered.

Examples of **FSDs** are:

- HPFS.IFS
- HPFS386.IFS
- CDROM.IFS

## (No title)

A **Driver Parameter Block (DPB)** contains vital information about the state and format of a disk drive. The DPBs are chained together and located in a single segment whose selector may be obtained from the [SAS](#). See [.A command](#)

See also related structures:

- [CDS](#)
- [MFT](#)
- [SFT](#)
- [FSC](#)
- [VPB](#)

Refer to the following for more detailed information

- [.A Kernel Debugger and Dump Formatter command.](#)
- [.D Kernel Debugger and Dump Formatter command.](#)

## (No title)

**Thrashing** refers to the state of a system where most of the CPU time is spent paging in and out memory from the swap file. This happens when real storage is heavily over committed and storage references encompass a wide range of virtual pages over a short processing time.

Such a condition can indicate a poorly tuned application where paging is caused by the process of accessing data the application needs. A typical scenario is where work data is chained in a single, very extended, queue and no mechanism exists to access the required data without scanning the entire chain. Use of hashing techniques greatly reduce this problem.

-----

## (No title)

The Compatibility Region Mapping Algorithm (also referred to as the *thunking* algorithm) is used by [thunking code](#) to convert 16:16 addresses to 0:32 addresses and *vice versa*.

This is achieved by ensuring [LDT](#) selectors have their limits set to 64K so that they *tile* the compatibility region (0M to 448M). This gives an easy conversion algorithm from the selector:offset address to the 32-bit linear address. In C language syntax this is expressed as follows:

```
linear_address=((selector >> 3) << 16) + offset  
selector:offset=((linear_address >> 13) | 7):(linear_address & 0x0000ffff)
```

-----

## (No title)

**Thunking** is the process of calling 16-bit code from 32-bit code and *vice versa*. Thunking consists of applying the [CRMA](#) to convert from one form of address to the other and making any stack parameter adjustments either by padding 16-bit operands to 32-bit with leading zeros (16- to 31-bit conversion) or truncating the padded 32-bit value to 16 bits (32- to 16-bit conversion).

-----

## (No title)

The **Translation Lookaside Buffer (TLB)** is a hardware implemented buffer used for caching linear to physical address mappings.

The Intel486(TM) processor provides **test registers** for manipulating the TLB.

-----

## (No title)

**cbargs** is the argument count associated with the hardware defined call [gate](#) mechanism. The count is the number of words or double-words (as defined by the gate descriptor) that are inserted into a ring 0 stack when ring 2 or ring 3 code executes a call gate instruction.

-----

## (No title)

A **paragraph** is a unit of memory allocation of 16 bytes. Paragraph aligned allocations lie on a 16-byte boundary.



---

## (No title)

The Program Data Block is the name given to the DOS [PSP](#) by the [DEM](#) component of OS/2.

---

## (No title)

The **Program Segment Prefix (PSP)** is a **DOS** control block that forms the header of a loaded program. Under OS/2 the [DEM](#) component refers to this as the PDB or Program Data Block.

---

## (No title)

The Loader Cache is used for saving discardable pages of instance data segments from DLLs loaded from mountable media. The caches is allocated from the kernel heap and has a system object owner ID of **cache**

---

## (No title)

The **Scheduler** component of OS/2 is responsible for managing [threads](#) on queues according to priority and status.

Refer to the following for more detailed information

[.P Kernel Debugger and Dump Formatter command.](#)

[.PB Kernel Debugger and Dump Formatter command.](#)

[.PU Kernel Debugger and Dump Formatter command.](#)

[.PQ Kernel Debugger command.](#)

---

## (No title)

A **Screen Group** is a logical full screen buffer and keyboard. A number of processes may be assigned to run in a given screen group. The workplace shell is one such screen group. Each screen group is assigned an ID. The screen group assigned to a process is recorded in its [Local Information Segment](#). The currently active screen group is recorded in the [Global Information Segment](#).

Screen Groups are represented by [SGCB](#) structures.

Under version 2 of OS/2 the screen group concept has been extended to that of a [session](#).

Refer to the following for information on displaying screen group ids.

[.P Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

The **Screen Group Control Block (SGCB)** is used by the session manager component of the system to represent a [Screen Group](#). It contains status information for the screen group and acts as a cross reference between the [Pid](#) currently associated with a given screen group and *vice versa*.

-----

## (No title)

**Sessions** are groups of related processes initiated using **DosStartSession** API. Each session is assigned a logical screen buffer or presentation space. Sessions are identified by a unique ID that corresponds with their [Screen Group](#) Id (though the range of numbers is extended to included PM sessions, which all share then same screen group).

The following session ID/Screen Group ID ranges are defined:

<i>SG</i>	<i>Usage</i>
0	Hard Error Popups
1	Shell Screen Group
2	Real Mode Screen Group
3	VioPopUp Screen Group
4	First Full Screen Application Session
15	Last Full Screen Application Session
16	First Windowable VIO-Session
31	Last Windowable VIO-Session
32	First PM session
255	Last PM session

-----

## (No title)

A **process** is a collection of threads that share a common address space.

Each process is primarily represented by a [PTDA](#) structure. and is assigned a unique identifier, the [Pid](#).

Processes are organised in hierarchical tree structures known as process or [Command Subtrees](#)..

-----

## (No title)

A **thread** is a independently scheduleable entity that competes for processor resource with other threads.

Each thread is represented by a [TCB](#) and [TSD](#) structure..

Threads are organised within processes and assigned a unique identifier within the owning process known as the [Tid](#).

All threads within the system are assigned a system wide unique identifier known as the [Thread Slot Number](#)

Refer to the following for more detailed information

[.P Kernel Debugger and Dump Formatter command.](#)

-----

## (No title)

The **Thread Slot Number** is a system wide unique identifier assigned to each thread in the system.

Threads are located from the thread slot table whose linear address is at global symbol:

`_papTCBSlots`

Each slot is a double-word linear address of the corresponding thread's TCB. The first slot (slot=0) is reserved.

Under the Kernel Debugger and Dump Formatter the following symbols may be used to represent particular threads in many of the commands that accept a slot number as a parameter:

\*                      The current or last dispatched thread as recorded in word global variable `_TaskNumber`

#                      The default thread slot used by the Dump Formatter and Kernel Debugger.

Refer to the following for more detailed information

[.P Kernel Debugger and Dump Formatter command.](#)

-----

## (No title)

The **Thread Identifier (tid)** is a value, unique within the owning processes, used to identify the thread. It is not the same as the [Thread Slot Number](#), which uniquely identifies a thread, system-wide.

The Tid is used in thread related APIs such as **DosKillThread** and **DosSetPriority**.

-----

## (No title)

The **Process Identifier (pid)** is a unique system wide value used to identify a given process.

**Note:** It is not the same as the [hptda](#) which

also uniquely identifies a process.

The Pid is used as a handle in process related APIs such as **DosKillProcess** and **DosWaitChild**.

Refer to the following for more detailed information

[.P Kernel Debugger and Dump Formatter command.](#)

-----

## (No title)

The Command Subtree Identifier is used to represent a part of a process (or command) tree headed by a particular parent process. The ID used is the [Pid](#) of the process that heads the subtree.

Normally a process has a CSID equal to it's own [Pid](#). However, when processes become orphaned they acquire the subtree Id of their original parent and become adopted by their grand-parents by acquiring their grand-parents's Pid as their new parent Pid.

Refer to the following for more detailed information

[.P Kernel Debugger and Dump Formatter command.](#)

## (No title)

A **Block Management Package (BMP)** is a generalised facility for managing tables of fixed length records. The BMP manages the commitment and decommitment of table storage and maintains a free a record list .p.The table is prefixed with a header structure (the [VMBH](#)) which facilitates the management of the BMP.

Typical examples of BMPs are:

- The [VMOB](#) table.
- The [VMAL](#) table.
- The [VMAR](#) table.

## (No title)

A **Record Management Package (RMP)** is a data structure designed for tabulating variable length records. Typically OS/2 uses **RMPs** to manage:

- Named Storage names
- Open File names
- Directory names
- System Semaphore names

## (No title)

**DEM** is the DOS Emmulation component of OS/2.

## (No title)

A **Page Table Entry (PTE)** is a hardware architected structure that is used to map virtual addresses to physical storage addresses.

Refer to the following for more detailed information

[DP Kernel Debugger and Dump Formatter command.](#)

## (No title)

The **User Virtual (UVIRT)** attribute signifies a virtual storage mapping to physical storage made without the full set memory management structures.

The UVIRT attribute may be associated with a number of structures, for example:

[PTE](#)

[LDT](#) and [GDT](#) descriptors.

[VMAL](#)

In general UVIRT allocations are 'convenience' mappings, which map memory to selector allocation. The attribute is used by Memory Management to signal minimal processing requirements. Typically they are created by device drivers using the `DevHlp_PhysToUvirt` facility. They are also created by the system where a 'quick' form of aliasing is required without the alias being associated with any memory object. Examples of this are:

VDM private arena memory

Selectors 5b and 53, 32-bit Code and Data selectors which are used by most 32-bit application programs.

Selector 30, the TASKAREA selector, which aliases the current process' [TSD](#), [TCB](#) and [PTDA](#).

-----

## (No title)

A **Virtual Page Structure (VP)** is used by memory management to track the status of a virtual storage frame, whether backed by physical storage, cached by the loader or paged out to the swapper. The Virtual Page Structures are allocated in contiguous storage, anchored from the address specified in global variable:

`_pgpVPBase`

Refer to the following for more detailed information

[.MV Kernel Debugger and Dump Formatter command.](#)

[VP](#) control block format

-----

## (No title)

A **Page Frame Structure (PF)** is used by page frame management to track the status of a physical storage frame. The Page Frame Structures are allocated in contiguous storage, anchored from the address specified in global variable:

`_pft`

Each PF corresponds one to one with a frame of physical storage and provides links to Virtual Page Structures ([VPs](#)).

Zero or more PTEs may be pinned to a physical frame, this is reflected in a reference count maintained in the associated PF.

[UVIRT](#) mappings have their corresponding PFs reserved unless aliased by non-UVIRT storage.

Refer to the following for more detailed information

[.MP Kernel Debugger and Dump Formatter command.](#)

**PF** control block format

---

## (No title)

The **Global Descriptor Table (GDT)** is a hardware architected control block. The **GDT** is common to all protect mode processes. It contains descriptors for memory segments common to all protect mode processes.

Refer to the following for more detailed information

[DG Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

The **Local Descriptor Table (LDT)** is a hardware architected table of memory descriptors.

Under OS/2 one LDT is allocated per process except for VDMs in which multiple LDTs are possible. A VDM will allocate one LDT per DPML client, but if it does not use DPML then the LDT is not initialised. When multiple LDTs are used in a VDM then the LDTR descriptor contents are updated to make an LDT current. This allows the same LDTR selector may be use regardless of current client.

Refer to the following for more detailed information:

[Kernel Debugger and Dump Formatter DL command.](#)

---

## (No title)

A **Virtual Memory Context Record (VMCO)** is used to record the association of shared arena, shared data objects with processes that are using.

Each **VMCO** is identified by a unique handle referred to as the [hco](#).

Refer to the following for more detailed information

[.MC Kernel Debugger and Dump Formatter command.](#)

**VMCO** control block format

---

## (No title)

The **Virtual Memory Object Record (VMOB)** are used to represent memory objects, that is the instance data associated with a particular virtual address. **VMOBs** contain pointers to the the owning and requesting objects as well as the corresponding [arena record \(VMAH\)](#).

Each VMOB is identified by a unique handle referred to as the [hob](#).

Refer to the following for more detailed information

[.MO Kernel Debugger and Dump Formatter command.](#)

**VMOB** control block format

---

## (No title)

The **Per-Task Data Area (PTDA)** is the anchor point for all process (task) related control information. One PTDA exists per process and from it is located the [LDT](#), [TCB](#) chain, Page tables and [Arena Headers](#) for a process.

All active PTDA's are addressable, whatever the current process, from a global address in the system arena. However, for the current process an alias address is created using selector 30 and in addition the many of the PTDA field names are declared as public symbols. This allows the fields names in the PTDA for the current process to be referred to directly under the Kernel Debugger and Dump Formatter.

**PTDA\_Start** is the symbol assigned to the beginning of the current **PTDA**. Using the [? command](#) against this and other PTDA field names allows relative offsets for PTDA fields to be calculated and used in other [contexts](#) as offsets from the global PTDA address.

Refer to the following for more detailed information

[.P Kernel Debugger and Dump Formatter command.](#)

**PTDA** control block format

---

## (No title)

One **Virtual Memory Arena Header Record (VMAH)** is allocated per arena to record information about the address range of an arena. The **VMAH** points to its sentinel arena record ([VMAR](#)).

Each VMAH chained in a double linked list.

The system arena **VMAH** is located at global symbol:

`_ahvmSys`

The shared arena **VMAH** is located at global symbol: `_ahvmShr`

For each private arena the **VMAH** is imbedded in the [PTDA](#) at label `ptda_ah` (`PTDA+0x40`).

Under OS/2 2.1 the system and shared arena **VMAHs** are assigned to objects 4 and 5 respectively.

Refer to the following for more detailed information

**VMAH** control block format

---

## (No title)

The **Virtual Memory Alias Record (VMAL)** is used to represent aliased regions of virtual memory. These are either:

regions of physical storage that may be addressed by more than one virtual, or

linear address that are not associated with a memory object, such as VDM [UVIRT](#) allocations.

When two memory objects are aliases of each other then they need not have co-incident sizes or origins within the aliased arena record. Aliases are designed to provide alternative attributes for accessing the same piece of data within or across processes. Compare this with shared instance data within the shared arena, where multiple [object records](#) share a common arena record. In this case each object is associated with a unique process and is not considered an alias.

Each **VMAL** is identified by a unique handle referred to as the [hal](#).

Refer to the following for more detailed information

[.ML Kernel Debugger command](#).

**VMAL** control block format

---

## (No title)

The **Virtual Memory Kernel Heap (VMKH)** structures are used to describe system heap memory. Many objects allocated out of the kernel heap are assigned a [System Object](#) identifier.

Refer to the following for formats of the kernel heap structures:

[VMKH - Kernel Heap Header](#)

[VMRKH - Resident Kernel Heap Structures](#)

[VMSKH - Swappable Kernel Heap Structures](#)

---

## (No title)

The **Virtual Memory Arena Record (VMAR)** is used to represent a contiguous region of virtual memory allocated in page quantities. Such storage may or may not be committed or resident.

Arena records are chained in a doubly linked lists, one for each arena type. That is, the chain chain exists separately for each private arena, the shared arena and system arena.

Special arena records, known as **Sentinels** head each chain. They describe the entire arena which they head.

All virtual memory is described by by at least one arena record.

Each **VMAR** is identified by a unique handle referred to as the [har](#).

Arena also records point to the following related memory structures:

[VMOB](#)

[VMAL](#)

[VMCO](#)

Refer to the following for more detailed information

[.MA Kernel Debugger and Dump Formatter command](#).

**VMAR** control block format

---

## (No title)

An **Object Table Entry (OTE)** describes the address, size and attributes an object within a loaded 32-bit load module.

The corresponding control block for a 16-bit load module is the [STE](#).



Refer to the following for more detailed information

[.LM Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

**A Segment Table Entry (STE)** describes the address, size and attributes of a segment (object) within a loaded 16-bit load module.

The corresponding control block for a 32-bit load module is the [OTE](#).

Refer to the following for more detailed information

[.LM Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

**The (non-swappable) Module Table Entry (MTE)** for a loaded module is use to record information about loaded modules. Since the **MTE** is allocated in non-swappable only information that must be resident at all times is recorded here. Related information that may be paged out is recorded in its sister control, the [Swappable Module Table Entry \(SMTE\)](#).

The MTE contains the following information:

- pointers to related control blocks such as: SMTE, resource and fix-up tables;
- attributes of the load module;
- Use count for .EXE modules.

Each MTE is identified by a unique handle referred to as the [hmte](#).

Refer to the following for more detailed information

[.LM Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

**The Swappable Module Table Entry (SMTE)** contains characteristics of a loaded module that may be page out of memory. The **SMTE** is the sister control block to the [MTE](#), which records those characteristics that must be resident at all times.

The SMTE principally contains:

- A pointer to [OTE](#) or [STE](#).
- A pointer to the fully qualified module name.
- The entry point and initial stack pointers.

Refer to the following for more detailed information

[.LM Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

The **CDA** is the Common ABIOS Data Area.

Refer to the following for more detailed information

[.C Kernel Debugger and Dump Formatter command.](#)

-----

## (No title)

The **Tread Control Block (TCB)** contains per-thread control and status information that must be resident at all times. The swappable counterpart to the TCB is the **TSD**

Refer to the following for related information

[.P Kernel Debugger and Dump Formatter command.](#)

**TCB** control block format

-----

## (No title)

The **Thread Swappable Data (TSD)** control block contains per-thread status and control information that resides in swappable memory and therefore is not required for reference out of context of the related thread. The resident memory counterpart to the **TSD** is the **TCB** (Thread Control Block).

The vast majority of the **TSD** is used as the ring 0 stack when a thread makes a privilege level transition to ring 0 via a [call gate](#) descriptor. The base of the ring 0 stack will therefore include the ring 3 call gate stack frame on entry to ring 0 (which is usually kernel or device driver code).

In the debug kernel a dummy page prefixes the used part of the TSD in order to catch ring 0 stack faults.

Other information contained in the TSD includes GTD instance data for the corresponding thread's [context](#). This comprises descriptors for:

28: The LDT descriptor.

30: Base selector for ring 0 process instance data, which includes the ring 0 stack, TCBs and PTDA.

38: Floating point emulator instance data

40: FS mapping to the TIB

When an inter-process thread context switches, descriptors 30 - 40 are loaded into the **GDT** from the TSD. When an intra-process thread context switches, descriptors 28 - 40 are loaded into the GDT from the TSD.

Refer to the following for related information

[.P Kernel Debugger and Dump Formatter command.](#)

**TSD** control block format

-----

## (No title)

The memory alias record handle (**hal**) is an index into the table of memory alias records (**VMALs**) whose address is located at **\_parVMAliases**.

Refer to the following for more detailed information

[.ML Kernel Debugger command.](#)

-----

## (No title)

A **hco** is a handle for a memory context record. This is an index into the table of memory arena records (**VMCOs**) whose address is located at **\_pcovmOne**.

Refer to the following for more detailed information

[.MC Kernel Debugger and Dump Formatter command.](#)

-----

## (No title)

The memory arena record handle (**har**) is an index into the table of memory arena records (**VMARs**) whose address is located at **\_parvmOne**.

Refer to the following for more detailed information

[.MA Kernel Debugger and Dump Formatter command.](#)

-----

## (No title)

The memory object record handle (**hob**) is an index into the table of memory objects records (**VMOBs**) whose address is located at **\_pobvmOne**

Refer to the following for more detailed information

[.MO Kernel Debugger and Dump Formatter command.](#)

-----

## (No title)

**Block-Ids** are conventional tokens used to represent the reason for a thread blocking. This occurs as the result of the kernel entering **TKSleep** (either directly or via **ProcBlock**). The address of the block-id is passed to **TKSleep** and stored in **TCBSleepID**. A thread wakes when the kernel calls **TKWakeup** (or **ProcRun**) with a corresponding block-id. All, zero or the highest priority thread blocked on the block-id will be woken depending on parameter flags. This mechanism is used by most functions and APIs that cause thread execution to be suspended, either for an event or serialisation.

Examples are:

**DosSleep**

DosSemWait  
DosWaitChild  
DosRead  
DevHlp\_ProcBlock

Refer to the following for more detailed information

[.PB Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

The [PTDA](#) control block handle **hptda** is the [hob](#) of the memory object that contains the **PTDA**.

**PTDAs** are allocated as [pseudo-objects](#), so do not have [Arena Records](#) associated with them.

Refer to the following for more detailed information

[.MO Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

The [MTE](#) control block handle (**hmte**) is the [hob](#) of the memory object that contains the **MTE**.

**MTEs** are allocated as [pseudo-objects](#), so do not have [Arena Records](#) associated with them.

Refer to the following for more detailed information:

[.MO Kernel Debugger and Dump Formatter command.](#)

[.LM Kernel Debugger and Dump Formatter command.](#)

---

## (No title)

**Pseudo-Objects** are small system objects that comprise control blocks and other system areas, which for reasons of virtual memory conservation are not represented by a corresponding [Arena Records](#). They are allocated out of the kernel resident heaps and comprise the following types of object:

[MTE](#)  
[VMAH](#)  
[PTDA](#)  
[Loader Cache](#)

Refer to the following for more detailed information

[.MO Kernel Debugger and Dump Formatter command.](#)

---

